Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Mediatekniikan suuntautumisvaihtoehto

2014

Ville Rantapuska

KOSKETUSNÄYTTÖOHJAUS ANDROID-MOBIILIPELISSÄ

– PenQ Rocket

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Mediatekniikka

Kesäkuu 2014 | 49 sivua

Ohjaaja: Yliopettaja, FM Mika Luimula

Ville Rantapuska

KOSKETUSNÄYTTÖOHJAUS ANDROID-MOBIILIPELISSÄ

Opinnäytetyön tarkoituksena oli toteuttaa kosketusohjattava mobiilipeli Android-laitteelle. Työtä varten perehdyttiin sekä kosketusnäyttöjen että mobiililaitteiden ominaisuuksiin ja toimintaperiaatteisiin. Tutkimuksessa tarkasteltiin erityisesti kosketusnäyttöjen hyöty- ja haittapuolia mobiilipelin ohjaimena. Havaittiin, että kosketusnäyttö on peliohjaimena helppokäyttöinen kunhan laitteen rajoitukset on otettu huomioon pelin suunnitteluvaiheessa. Näitä tietoja hyödynnettiin mobiilipelin rakenteen, ulkonäön ja käyttöliittymän toteutuksessa.

Työssä vertailtiin seitsemää mobiilipelin kehitykseen soveltuvaa pelimoottoria ja pelinkehitysympäristöä. Vertailuun valittiin mahdollisimman erilaisia vaihtoehtokokonaisuuksia. Tuloksissa painotettiin erityisesti nuorelle startup-yritykselle olennaisia ominaisuuksia. Havaittiin että markkinoilta löytyi runsaasti varteenotettavia mobiilipelimoottorivaihtoehtoja. Kaikki Android-laitteilla toimivat pelimoottorit tukivat myös kosketusnäyttöohjausta. Pelimoottorivertailun tuloksena Unity valittiin Android-mobiilipelin kehittämiseen parhaiten soveltuvaksi kehitysympäristöksi.

Käytännön osuutena toteutettiin PenQ Rocket -ongintaseikkailupelin prototyyppi, joka tuotiin Android-mobiililaitteelle.

ASIASANAT:

Android, kosketusnäyttö, mobiilipelit

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Digital media

June 2014 | 49 pages

Instructor: Principal Lecturer, Ph. D. Mika Luimula

Ville Rantapuska

TOUCHSCREEN CONTROL OF AN ANDROID MOBILE GAME

The goal of this thesis was to create a touchscreen-controlled mobile game for Android devices. For this purpose both touchscreen and Android mobile devices were studied. The study focused on benefits and challenges of using a touchscreen as a mobile game controller. It was found that the touchscreen is a proper and easy to use controller option for games as long as its limitations have been taken into account during the development of the game. These findings were used for the creation of the structure, appearance, and command interface of the game.

Seven different mobile game engines and software development kits were compared. Items for the comparison were selected so that they include a diverse range of software. Features relevant to a young start-up company were emphasized. It was found that there are many noteworthy mobile game engines on the market. All game engines that supported Android operating system also supported touchscreen control to some extent. In the end, Unity was chosen as most suitable game engine, and was used in the implementation of the project.

A concrete result of the project was a fishing adventure game prototype for Android devices.

KEYWORDS:

Android, game, mobile, touchscreen

SISÄLTÖ

LIITTEET

KUVAT

TAULUKOT

SANASTO

2D	Kaksiulotteinen (2 dimensional).
3D	Kolmiulotteinen (3 dimensional).
ADT	Kokoelma Android-sovelluskehitykseen tarvittavia työkaluja (Android Developer Tools).
Android-laite	Kännykkä tai tablettitietokone joka perustuu Google Android -ohjelmistopinoon ja -käyttöjärjestelmään.
C++	C-ohjelmointikielestä laajennettu versio. C++ on maailman tärkeimpiä ohjelmointikieliä etenkin suoritustehokkuudesta puhuttaessa.
C#	C# (C sharp) on Microsoftin kehittämä ohjelmointikieli. Se yhdistää C++:n tehokkuuden ja Java-kielen helppokäyttöisyyden.
Kasuaalipeli	Kasuaalipeli (casual game) on pieni ajanvietepeli, joka on helppo aloittaa ja lopetaa.
Mobiilipeli	Liikuteltavalle laitteelle suunniteltu digitaalinen peli.
Pelimoottori	Pelimoottori (engine) on tietokonepelin pohjana toimiva ohjelmistokokonaisuus.
Prototyyppi	Tuotteen esiversio tai mallikappale.
px	Pikseli (pixel) eli kuvapiste näytön ruudulla.
Resoluutio	Näyttöruudun tarkkuus eli pikselimäärä. Ilmaistaan useimmiten muodossa leveys x korkeus.
SDK	Kehitysympäristö eli kokoelma sovelluskehitykseen tarkoitettuja ohjelmia (Software Development Kit).
SVGA	Eräs näyttöjen standardiresoluutio (Super Video Graphics Array) kooltaan 800 x 600 pikseliä.
Tabletti	Taulutietokone eli litteä ja yksiosainen kannettava tietokone, jota ohjataan usein kosketusnäytön avulla.
WVGA	Eräs standardiresoluutio (Wide Video Graphics Array), mitoiltaan 800 x 480 pikseliä.
WXGA	Eräs epästandardi resoluutio (Wide Extended Graphics Array), tyypillisesti 1 366 x 768 tai 1 280 x 800 pikseliä.
WQXGA	Eräs standardiresoluutio (Wide Quad Extended Graphics Array), kooltaan 2 560 x 1 600 pikseliä.

1 JOHDANTO

Tässä opinnäytetyössä tutustutaan pelisovelluksen toteuttamiseen Android-mobiililaitteelle. Työssä keskitytään erityisesti kosketusnäytön käyttöön peliohjaimena sekä sen tuomiin mahdollisuuksiin ja haasteisiin. Tämän uuden ohjaintyypin toiminnan ymmärtämiseksi tekstissä käydään läpi kosketusnäyttöihin liittyvää teoriaa sekä erilaisia kosketusnäyttötyyppejä.

Tekstissä tutkitaan lyhyesti nykyisten mobiililaitteiden ominaisuuksia ja suorituskykyä. Päätelaitteen ominaisuudet ja rajoitukset on otettava huomioon jo pelin suunnitteluvaiheessa. Mobiililaitteiden suorituskyvyn hahmottaminen edesauttaa toteutusvaiheen sulavuutta.

Kehitystyön nopeuttamiseksi pelin pohjaksi etsitään markkinoilta yksi pelimoottori, jonka päälle varsinainen sisältö toteutetaan. Työssä vertaillaan useita erilaisia pelimoottoreita ja valitaan niistä Android-mobiilipelille soveltuvin vaihtoehto.

Opinnäytetyön osana tullaan toteuttamaan prototyyppi kosketusnäytöllä ohjattavasta mobiilipelistä, PenQ Rocketista. Toimeksiantajana peliprojektille toimii Pinniini Design -niminen nuori startup-yritys.

Työ päätetään pohdintaan kosketusnäytön toimivuudesta peliohjaimena sekä PenQ Rocket -pelin jatkokehitysmahdollisuuksista.

2 KOSKETUSNÄYTTÖ

Kosketusnäyttö on elektroninen näyttöruutu, jonka kautta käyttäjä sekä saa että syöttää tietoa. Tyypillisin tapa tiedon syöttämiseen kosketusnäytöltä on näytön painaminen sormenpäällä. Vaihtoehtoisesti näyttöä voidaan koskettaa tähän tarkoitukseen valmistetulla osoitinkynällä (stylus). Osoitinkynä on hieman sormipainallusta tarkempi tiedonsyöttötapa, mutta lisää järjestelmään yhden helposti hukkuvan osan. Kehittyneemmät kosketusnäytöt kykenevät tunnistamaan useita samanaikaisia kosketuksia. Näitä laitteita kutsutaan monikosketusnäytöiksi. Kosketusnäytön tyyppi ja ominaisuudet riippuvat laitteen käyttötarkoituksesta ja -ympäristöstä [1].

Esineiden manipulointi koskettamalla on ihmiselle luonnollinen tapa vaikuttaa ympäristöönsä. Tätä tapaa on käytetty aikojen alusta asti, aina ovenrivoista valokatkaisijoihin. 2000-luvulla yleistyneet kosketusnäytöt [2] jatkavat tätä perinnettä ja mahdollistavat entistä intuitiivisemman vuorovaikutuksen ihmisen ja digitaalisen maailman välillä.

2.1 Historia

Vuonna 1965 E. A. Johnson tutkimusryhmineen julkaisi tuloksensa kapasitiivisista kosketusnäyttöruuduista. Muutamaa vuotta myöhemmin toteutettiin ensimmäiset kosketusnäytöt, joita käytettiin hyödyksi lentoliikenteen ohjaamisessa. Näitä laitteita (kuva 1) ei valmistettu laajaan kuluttajakäyttöön, vaikka ne koettiin toimiviksi. Samaa kapasitiivista tekniikkaa hyödynnetään kuitenkin edelleen moderneissa kosketusnäytöissä. [3]

Kuva 1. E. A. Johnsonin kapasitiivinen kosketusnäyttö vuodelta 1968 [3].

Ensimmäinen kaupallinen kosketusnäyttöä hyödyntävä tietokone, HP-150 (kuva 2), saapui markkinoille vuonna 1983. Laitteesta löytyi näppäimistön tueksi infrapunalla toimiva kosketuksen tunnistava näyttöpääte. Infrapunasensorit oli sijoitettu kuvaputkimonitorin reunoille. Sormen tai muun läpinäkymättömän esineen lähestyessä ruutua infrapunasäde katkesi, jolloin kosketuksen sijainti voitiin tunnistaa. [4]

Kuva 2. Ensimmäinen kaupallinen kosketusnäyttötietokone, HP-150 [5].

Monikosketusnäyttöjen ensimmäiset prototyypit valmistettiin 1983 ja ne toimivat näytön taakse sijoitetun kameran avulla. Kamerajärjestelmä oli tilaavievä, joten tutkijat etsivät vaihtoehtoisia tekniikoita. Vuonna 1985 Toronton yliopistossa kehitettiin tablettitietokone, joka perustui kapasitiiviseen kosketusnäyttöön ja kykeni monikosketukseen [6].

Monikosketusteknologian myötä kehitys jatkui erilaisten eleiden tunnistamistekniikoilla. Vuonna 1990 Andrew Sears työryhmineen julkaisi tutkimuksen eleistä ja tarkkojen kosketusnäyttöjen uudesta aikakaudesta [7]. Tämä aloitti kosketusnäytöllä toimivien mobiililaitteiden nousun.

2.2 Teknologiat

Kosketusnäyttö voidaan toteuttaa lukuisin eri tekniikoin. Yleisimmät näistä ovat kapasitiivinen sekä resistiivinen toimintaperiaate, mutta kosketuksen tunnistusta on testattu myös näkyvään valoon, infrapunasäteilyyn sekä ääniaaltoihin perustuvilla tekniikoilla. Tässä luvussa käydään läpi mobiililaitteiden yleisimmät kosketusnäyttöteknologiat.

2.2.1 Kapasitiivinen

Kapasitiivinen kosketusnäyttö koostuu lasisesta eristekerroksesta, jonka sisäpinnalla on sähköä johtava kerros. Ihmisen iho johtaa sähköä, joten sormen vieminen näytön lähelle vaikuttaa näytön alemman kerroksen magneettikenttään. Tämä muutos voidaan havaita ja paikantaa näytöllä. Muutoksen havaitseminen perustuu magneettikentän muutokseen, mikä tarkoittaa, että kapasitiivista kosketusnäyttöä voi ohjata ainoastaan sähköä johtavalla osoittimella. [8]

Kapasitiivinen kosketusnäyttö on yleisin mobiililaitteissa käytetty kosketusnäyttötyyppi. Ne ovat erittäin tarkkoja ja reagoivat kosketukseen

välittömästi. Kapasitiivinen teknologia mahdollistaa myös useamman kosketuksen tunnistamisen yhtäaikaisesti eli niin kutsutun monikosketusnäytön.

2.2.2 Resistiivinen

Resistiivinen näyttöteknologia perustuu kahteen tai useampaan vastakkain asetettuun kalvoon, jotka johtavat sähköä. Näiden kalvojen välissä on ohut eristekerros. Käyttäjän koskettaessa näyttöä sähköä johtavat kalvot painuvat lähemmäs toisiaan. Kosketusvoiman ollessa tarpeeksi suuri sähkövirta pääsee kulkemaan eristekerroksen läpi. Syntynyt virtapiiri voidaan havaita ja paikallistaa näytöltä mikroprosessorin avulla. [9]

Resistiivisiä kosketusnäyttöjä voi käyttää millä osoittimella tahansa. Näytön rakenteen takia osoittaminen vaatii jonkin verran painetta, jotta kosketus rekisteröityy. Tarkkuudeltaan resistiiviset kosketusnäytöt ovat hiukan kapasitiivisia näyttöjä epätarkempia, mutta niiden valmistusprosessi on edullisempi [9]. Pienempien tuotantokustannusten vuoksi resistiiviset kosketusnäytöt ovat varsin suosittuja valmistajien keskuudessa. Monikosketuksen toteuttaminen resistiivisellä näytöllä on teknisesti hankalaa.

2.3 Kosketuseleet

Kosketusnäytöstä puhuttaessa eleellä tarkoitetaan yhtenäistä osoittimella tehtävää liikerataa. Liikeradan muodon perusteella tietokoneohjelma määrittää mikä komento suoritetaan. Erilaisia liikeratoja voi olla lähes lukematon määrä, etenkin kun niihin yhdistetään monikosketusten mahdollisuus. On kuitenkin otettava huomioon ihmisen rajallinen kyky muistaa kosketuseleiden muotoja ja syötteen epätarkkuus toistettaessa niitä kädenliikkeinä. Parhaassa tapauksessa eleet tekevät tiedon syöttämisestä nopeaa ja intuitiivista, mutta huonosti toteutettuna ne pakottavat käyttäjän muistamaan ulkoa lukuisia epäloogisia pyyhkäisyjä. Tässä luvussa käydään läpi yleisimmät kosketuseleet: painaminen,

pyyhkaisy, raahaaminen, vierittäminen ja nipistäminen. Napautusta ei tavallisesti katsota eleeksi, koska siitä puuttuu eleelle tyypillinen liikerata. [10]

Painaminen ja painettuna pitäminen (tap and hold) on kosketuseleistä yksinkertaisin. Siinä osoitin asetetaan ruudulle, pidetään paikallaan ja irrotetaan ruudusta (kuva 3). Painaminen liitetään yleensä jonkin kohteen tutkimiseen, valintaan tai käsittelemiseen. [11] Tätä käytetään tyypillisesti hiiren oikean painikkeen tavoin tuomaan ruudulle ponnahdusvalikko (popup menu).

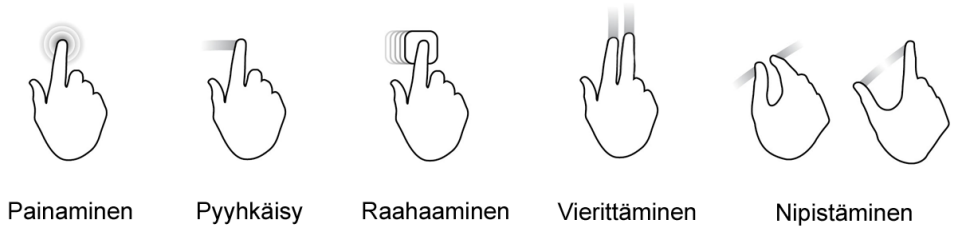
Pyyhkäisy tai liu'uttaminen (swipe) on eräs tunnetuimmista kosketuseleistä. Siinä osoitin asetetaan ruudulle, liikutetaan lineaarisesti johonkin suuntaan ja irrotetaan ruudusta (kuva 3). Näin luodaan varsin tutunomainen pyyhkäisyliike, jota ihmiset ovat ennestään tottuneet käyttämään esimerkiksi kirjan sivujen kääntämiseen. [11] Tätä elettä onkin loogista käyttää erilaisissa sivunvaihdoissa ja valikkosiirtymissä.

Raahaaminen (drag) muistuttaa pyyhkäisyä. Tässä osoitin asetetaan ruudulle jonkin kohteen päälle, sitä liikutetaan vapaasti ruudulla ja irroitetaan ruudusta (kuva 3). Tämä ele liitetään loogisesti raahattavan kohteen siirtämiseen paikasta toiseen. [11] Pyyhkäisystä eroten raahaaminen voi tapahtua hitaammin, eikä sen suunta ole ennalta määrätty. Joissain tapauksissa raahaaminen voi päättyä samaan kohtaan ruudussa mistä se alkoikin. Tällöin käyttöjärjestelmän tulee havaita ettei käyttäjä halunnutkaan siirtää kyseistä kohdetta vaan perui komennon. Raahaamista käytetään usein tiedostojen kopioinnissa ja siirtämisessä hakemistosta tai laitteesta toiseen.

Vierittäminen (pan) on yksinkertainen monikosketusele, joka muistuttaa pyyhkäisyä. Eleessä kaksi sormea asetetaan ruudulle vierekkäin ja niitä kumpaakin liikutetaan samaan aikaan vaaka- tai pystyakselin suuntaisesti (kuva 3). [11] Vierittämistä käytetään tyypillisesti ruutua suuremman dokumentin liikutteluun. Se vastaa läheisesti hiiren rullan toimintoja.

Nipistäminen (pinch) lienee monikosketuseleistä tunnetuin. Siinä kaksi sormea asetetaan ruudulle, liikutetaan lähemmäs tai etäämmäs toisistaan ja irrotetaan ruudusta (kuva 3). [11] Tämä ele voi aluksi tuntua epäloogiselta, mutta jos se

ajatellaan vaikkapa kankaan rypistämisenä ja venyttämisenä, voidaan ele yhdistää kohteen suurentamiseen ja pienentämiseen. Nipistyskomennon yleisin käyttötarkoitus onkin dokumenttien tai valokuvien lähennys- ja loitonnustoiminto (zoom).



Kuva 3. Yleisimmät kosketuseleet.

Edellisten lisäksi kosketusnäyttölaitteista löytyy usein valmistajakohtaisia kosketuseleitä, jotka eivät ole vielä vakiinnuttaneet asemaansa suuren yleisön keskuudessa. Mitä enemmän kosketuseleitä vakiintuu tällä tavoin sitä enemmän niitä voidaan käyttää myös pelisovelluksissa. Valmiiksi opitut kosketuseleet eivät vaadi pelaajalta uuden opettelua ja sopivat siltä osin myös pelien komennoiksi.

2.4 Kosketusnäyttö mobiililaitteessa

Mobiililaite asettaa kosketusnäytölle rajoitteita koon, virrankulutuksen ja kestävyyden suhteen. Näytön tulee olla pieni ja kevyt mutta myös energiatehokas. Näytön kirkkauden tulee olla riittävä, jotta laitetta voidaan käyttää suorassa auringonvalossa. Käyttöliittymäsuunnittelussa on huomioitava painikkeiden koko, jotta niihin on mahdollista osua sormella luontevasti myös laitteen ollessa liikkeessä. Painikkeiden väliin tulee jättää ylimääräistä tilaa virhepainalluksien vähentämiseksi [12]. Suurimmaksi ongelmaksi on kuitenkin osoittautunut näyttöjen kestävyys. Ihmisen mukana kulkevaan laitteeseen kohdistuu huomattavasti enemmän fyysisiä uhkia kuin pöydälle sijoitettuun

monitoriin. Vaurioitunut kosketusnäyttö onkin mobiililaitteiden ylivoimaisesti yleisin vika [13].

2.5 Kosketusnäyttö pelikäytössä

Kosketusnäyttö on helppo ja intuitiivinen tapa ohjata tietokonetta. Peleissä tämä kontrollimekaniikka aiheuttaa kuitenkin myös rajoituksia. Ohjaaminen kosketusnäytöllä on hitaampaa ja epätarkempaa kuin perinteisellä näppäimistöllä ja hiirellä. Vain harvoissa tablettitietokoneissa on mukana teline, joten käyttäjän täytyy pitää laitetta käsissään. Komentojen antamiseen jää siis käytettäväksi ainoastaan muutama sormi. [12] Tästä syystä mobiililaitteille suunniteltujen pelien käyttöliittymän tulee olla perinteistä tietokonepeliä yksinkertaisempi. Etenkin tekstimuotoisen syötteen vaatimista pelin aikana on vältettävä, sillä virtuaalinen näppäimistö peittää alleen suuren osan ruutua ja samalla pelimaailmaa.

Sormien käyttäminen osoittimena tuo kontrollimekaniikkaan sekä hyötyjä että haittoja. Toisaalta käyttäjä on aina tietoinen missä hänen sormensa ovat, joten osoitin ei huku ruudulla näkyvään sisältöön, toisin kuin vaikkapa hiiriohjauksessa on mahdollista käydä. Negatiivisena puolena voidaan nähdä päinvastainen ilmiö; ruudulla oleva sisältö voi helposti jäädä sormien alle piiloon [12]. Tätä kompensoidaan tyypillisesti suurentamalla pelimaailman esineiden grafiikan kokoa näyttöruudulla.

2.6 Tulevaisuus

Kosketusnäyttöteknologiat ovat kehittyneet tasaiseen tahtiin jo vuosikymmeniä, eikä ole syytä olettaa, että kehitys päättyisi nykyiselle tasolle. Kosketusohjausta on viime vuosina sovellettu useissa laitteissa, ja uusia tapoja löydetään jatkuvasti lisää. Taittuvat kosketusnäytöt ja sähköinen paperi tekevät jo tuloaan [14]. Tulevaisuudessa yleistyvät varmasti myös Microsoft Kinect -tyyppiset ratkaisut, joissa laitteita ohjataan pelkästään eleiden avulla ilman varsinaista

kosketusta [15]. Kolmiulotteisten näyttöjen kehittyessä tullaan lähitulevaisuudessa varmasti näkemään 3D-kosketusnäyttöjä ja hologrammeihin pohjautuvia ohjausratkaisuja.

Tactus Technologiesin vision mukaan kosketusnäyttöihin tuodaan takaisin perinteisen näppäimistön taktiili palaute. Yritys on kehittänyt näytön, jonka pinnanmuotoa voidaan dynaamisesti muokata. Täten ruudulle piirrettävän näppäimistön painikkeet voidaan kohottaa ulos muusta ruutupinnasta. Tutkimuksen mukaan näppäimistön painikkeiden tunteminen sormenpäillä lisää kirjoitusnopeutta ja vähentää syöttövirheiden määrää [16]. Pidemmälle vietynä tämä teknologia voisi mukauttaa näytön pinnan vastaamaan mitä tahansa ruudulla näkyvää sisältöä; esimerkiksi koiranpennun kuva tuntuisi pehmeän silkkiseltä.

Kehittyvien valmistustekniikoiden myötä kosketusnäyttöjen tuotantokustannusten voidaan olettaa laskevan, mikä mahdollistaa sekä entistä pienempien että entistä suurempien näyttöjen valmistuksen. Tulevaisuudessa vaikkapa huoneiston koko seinäpinta-ala voi olla kosketusohjattava näyttö. Tällöin taulujen ripustaminen onnistuisi muutamalla sormieleellä ja tapetit olisi mahdollista vaihtaa mielialan mukaan vaikka useamman kerran viikossa.

3 ANDROID-MOBIILILAITE

Mobiililaitteiden, eli kännyköiden ja tablettitietokoneiden, määrä kotitalouksissa on viimevuosina kasvanut huomattavasti [17]. Näiden pienten tietokoneiden menestys perustuu hyvään liikuteltavuuteen sekä helppoon käytettävyyteen. Kosketusnäytön myötä erillistä näppäimistöä ei tarvita, mikä pienentää laitteen kokoa entisestään. Langaton verkkoyhteys sallii vapaan liikkuvuuden. Mobiililaite kulkee helposti käyttäjän mukana eri ympäristöihin aina olohuoneen sohvalta telttaretkelle.

Android-mobiililaitteita on markkinoilla lukuisia eri malleja. Tässä osiossa käydään pikaisesti läpi laitteiden yhteiset piirteet ja ominaisuudet. Perustiedot pelialustasta ovat olennaisia pelinkehittäjille, sillä nämä ominaisuudet vaikuttavat suuresti pelikokemukseen ja rajaavat pelin käytettävissä olevat resurssit.

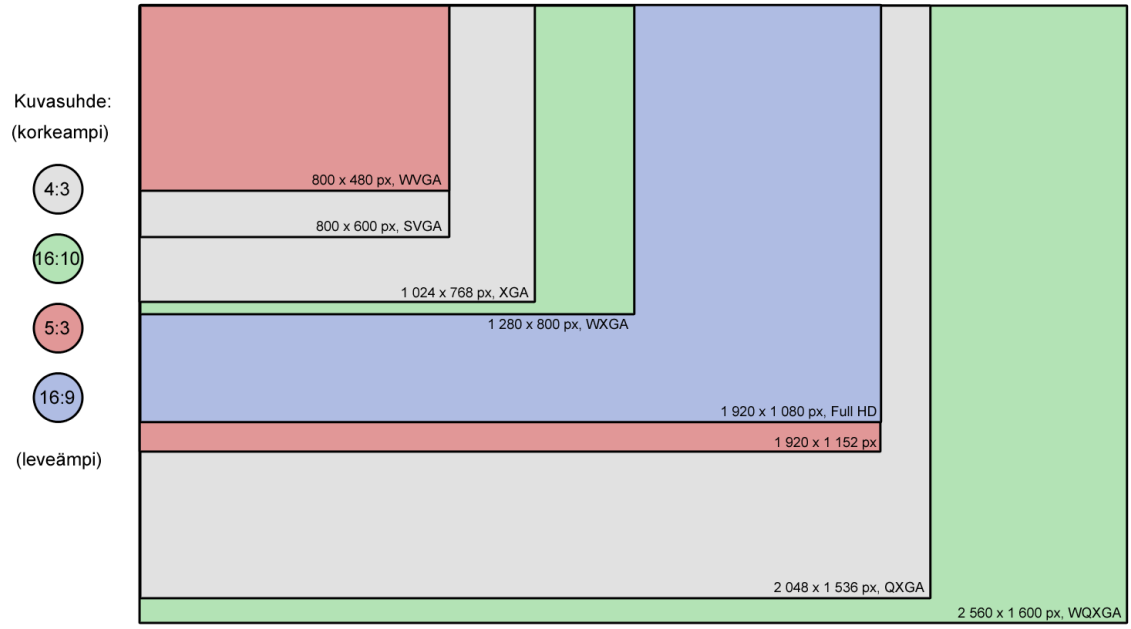
3.1 Android-käyttöjärjestelmä

Android on Linuxin avoimeen käyttöjärjestelmäytimeen perustuva käyttöjärjestelmä, joka julkaistiin vuonna 2007. Sen kehityksestä vastaavat Google ja Open Handset Alliance. Vuonna 2013 Android oli mobiililaitteiden yleisin käyttöjärjestelmä kattaen 81,3 % älypuhelinmarkkinoilla myydyistä tuotteista [18]. Käyttöjärjestelmän viimeisin versio 4.4.2 ”KitKat” julkaistiin joulukuussa 2013. Android suunniteltiin nimen omaan kosketusnavigoitavia mobiililaitteita varten, ja sen peruskomentoihin kuuluvat kosketuseleet, kuten luvussa 2.3 mainitut pyyhkäisyt, raahaaminen sekä nipistykset.

3.2 Näyttö

Android-laitteiden näyttöjen resoluutiot ja kuvasuhteet vaihtelevat mallin mukaan laadukkaimpien laitteiden WQXGA-resoluutiosta (2 560 x 1 600 px)

halvempien tuotteiden WVGA-resoluutioon (800 x 480 px) (kuva 4). Keskimäärin tablettitietokoneen resoluutio on WXGA-muotoa (1 280 x 800 px) ja kuvasuhde tyypillisesti 16 : 10 [19]. Nämä eivät vastaa televisioteollisuudesta tuttua FullHD-resoluutiota (1 920 x 1 080 px, 16 : 9), mikä on hyvä tiedostaa jo pelin suunnittelu- ja kehitysvaiheessa. [20]



Kuva 4. Android-mobiililaitteiden resoluutioita.

Resoluutioiden lisäksi näyttöjen pikselitiheys vaihtelee eri Android-laitteiden välillä. Pikselitiheydellä (dpi, dots per inch) tarkoitetaan sitä, kuinka monta kuvapistettä mahtuu yhden tuuman (2,54 cm) matkalle. Tämä on kääntäen verrannollinen näyttöpisteiden kokoon, eli mitä suurempi on pikselitiheys sitä pienempiä ovat pikselit. Android-laitteiden yleisimmät pikselitiheydet ovat:

- 120 dpi (low density)
- 160 dpi (medium density)
- 240 dpi (high density)
- 320 dpi (extra high density).

Matalia pikselitiheyksiä nähdään lähinnä vanhemmissa laitteissa. Keskitason tiheyksiä puolestaan tavataan nykypäivän edullisemmissa tuotteissa. Ammattitason laitteet ja kalliimmat trendituotteet sisältävät lähes poikkeuksetta korkean pikselitiheyden kosketusnäytön. Erilaisten pikselitiheyksien vuoksi peligrafiikoista joudutaan usein tekemään vaihtoehtoisia versiota, jotta grafiikan laatu ja koko ruudulla pysyvät siedettävinä. [20]

3.3 Suorituskyky

Suorituskyvyltään Android-laitteet vaihtelevat suuresti johtuen mobiililaitteiden nopeasta kehitystahdista. Uudet laitteet perustuvat useimmiten Qualcommin Snapdragon- tai Nvidian Tegra-alustoihin. Nämä ovat mikropiirikokonaisuuksia sisältäen kaksi- tai neliytimisen suorittimen, grafiikkapiirin sekä muistinhallinnan. Kellotaajuuksiltaan ne yltävät aina kahteen gigahertsiin asti ja ovat siten täysin verrattavissa kannettavien tietokoneiden suorituskykyyn. [21][22] Suurin ongelma kasvavassa suorituskyvyssä on tarvittavan jäähdytyksen määrä, koska pieneen tilaan rakennetut sähkölaitteet ylikuumenevat helposti etenkin pitkän yhtäjaksoisen käytön seurauksena. Estääkseen komponenttien vioittumisen mobiililaitteet tyypillisesti ajavat tehoaan dynaamisesti alaspäin, kun lämpötila lähestyy laitteelle asetettua ylärajaa. Etenkin laskennallisesti raskaissa 3D-peleissä tämä huomataan helposti. Mobiililaitteelle kasuaalipelit eli pienet ajanvietepelit ovatkin paras vaihtoehto: lyhyissä pelisessioissa laite ei ehdi lämmetä liiallisesti.

3.4 Liitettävyys

Android-laitteet kommunikoivat ympäröivien laitteiden kanssa pääasiassa langattomien yhteyksien kautta. SIM-korteilla varustettujen laitteiden on mahdollista liittyä matkapuhelinverkkoihin 3G- tai 4G-tekniikoiden avulla. Langattomaan lähiverkkoon (WLAN) laitteet kytkeytyvät IEEE 802.11

-standardin mukaisella yhteydellä (Wi-Fi). Bluetooth-ominaisuus laajentaa liitettävyyttä lyhyillä etäisyyksillä. [23]

Vanhemmista laitteista löytyy tyypillisesti infrapunalähetin, mutta uusista kokoonpainoista tämä vanhentuva teknologia on jätetty pois. Uudemmista laitteista vastaavasti löytyvät NFC (Near Field Communication) ja DLNA (Digital Living Network Alliance) -sertifikaatit. NFC mahdollistaa kahden laitteen välisen kättelyn hyvin lyhyillä, muutaman senttimetrin, etäisyyksillä RFID-tekniikkaa (Radio Frequency Identification) hyödyntäen. DLNA on UPnP-protokollia (Universal Plug and Play) hyödyntävä menetelmä, joka helpottaa digitaalisen median, kuten valokuvien ja videon, siirtämistä lähiverkkoon kytkettyjen laitteiden välillä. USB-kaapeli (Universal Serial Bus) on tyypillisesti ainoa fyysinen tapa liittää mobiililaite muihin laitteisiin. [23]

4 PELIMOOTTORIVERTAILU

Pelimoottorit ovat ohjelmistoalustoja, joiden päälle tietokonepelejä rakennetaan. Tietokonepelit ovat monimutkaisia ohjelmia ja niitä on epäkäytännöllistä toteuttaa jokaisella kerralla alusta asti, sillä pelien perusominaisuudet muistuttavat hyvin paljon toisiaan. Kehitystyötä nopeuttavat valmiit pelimoottorit, jotka sisältävät nämä yleisimmät peleissä käytettävät ominaisuudet. Näin pelinkehittäjät voivat keskittyä sisällöntuotantoon. Tässä osiossa vertaillaan tunnetuimpia pelimoottoreita [24] ja tutkitaan, mikä niistä sopii parhaiten kosketusnäytöllä ohjattavan Android-mobiilipelin pohjaksi [25]. [26]

4.1 Pelimoottori vai pelinkehitysympäristö

Ennen pelimoottorivertailua on tärkeää erottaa toisistaan pelimoottorit (engine) ja kehitysympäristöt (SDK). Pelimoottori on pelin ydin, joka hoitaa toimintaa pelin ollessa käynnissä. Pelinkehitysympäristö on puolestaan kokoelma työkaluja, joilla pelin sisältöä ja pelimoottorin käyttäytymistä voidaan muokata. Usein nämä kaksi toimitetaan samassa tiedostopaketissa käyttöönoton nopeuttamiseksi ja versioyhteensopivuuden varmistamiseksi. Aina näin ei kuitenkaan ole ja joillekin pelimoottoreille ei ole saatavilla omaa kehitysympäristöä lainkaan.

4.2 Haetut ominaisuudet

Vertailussa painotetaan Android-mobiilipeliprojektin kannalta tärkeitä ominaisuuksia. Pelimoottoreilta toivotut omainaisuudet listattiin tärkeimmästä vähemmän tärkeisiin. Listauksessa otettiin huomioon nuoren startup-yrityksen tarpeet:

- yhteensopivuus uusien Android-laitteiden kanssa
- kosketusnäyttöohjauksen tuki

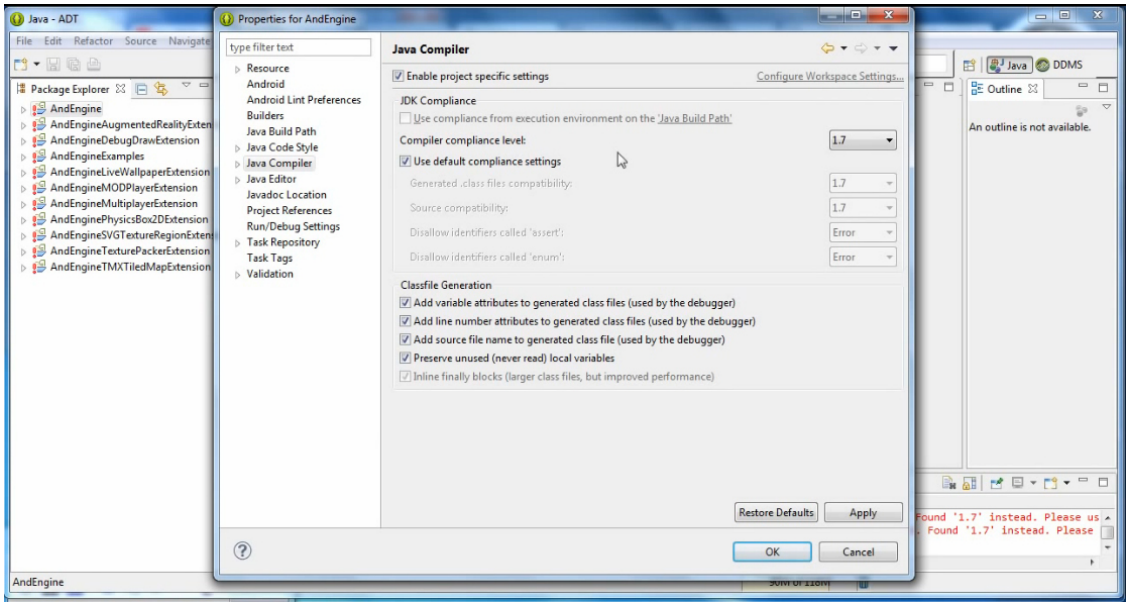
- kyky toteuttaa ja ajaa 2D-sovelluksia
- helposti opittava käyttöliittymä
- hinta
- moottorilla julkaistu onnistuneita pelejä
- tukipalvelut
- laajennettavuus muille laitteille.

Näiden ominaisuuksien perusteella valittiin seitsemän markkinoilla olevaa pelimoottoria tarkempaa vertailua varten. Vertailuun valitut pelimoottorit olivat AndEngine, GameMaker, GameSalad, Corona, Marmalade, Unity ja Unreal.

4.3 AndEngine

Nicolas Gramlichin kehittämä AndEngine on ilmainen 2D-pelimoottori Android-sovellusten luomiseen. Gramlich tunnetaan yhteistyöstä Facebook-pelejä julkaisevan Zynga-peliyhtiön kanssa. AndEngine perustuu sulautettujen laitteiden avoimeen grafiikkakirjastoon OpenGL for Embedded Systems (OpenGL ES) ja tukee kirjaston ensimmäistä sekä toista versiota: OpenGL ES 1 ja OpenGL ES 2 [27].

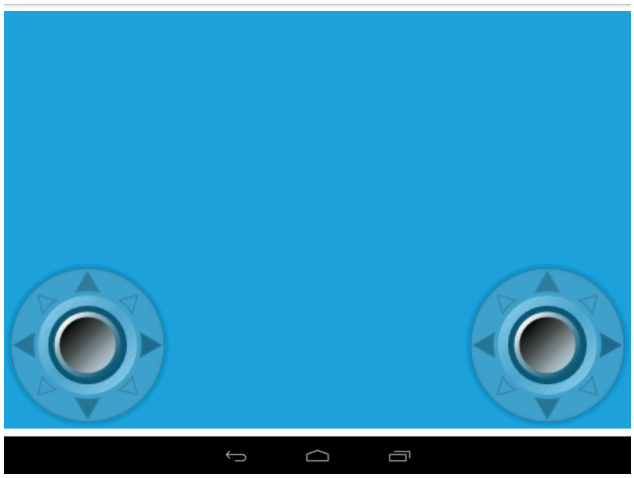
Moottori on toteutettu pääasiassa Java-ohjelmointikielellä, mutta sisältää myös muutamia C++ -ohjelmakirjastoja, kuten x86 natiivit kirjastot. Koko AndEnginen lähdekoodi on avoin ja ladattavissa GitHub-sivustolta osoitteesta https://github.com/nicolasgramlich/AndEngine. Nimensä mukaisesti AndEngine sisältää ainoastaan pelimoottorin eikä erillistä työkalua ohjelmistokehitykseen. Rinnalle tarvitaan siis erillinen kehitysympäristö, jonka avulla pelejä luodaan. Yksi tällainen ympäristö on Eclipse (kuva 5).



Kuva 5. Ruutukaappaus AndEnginestä Eclipse-kehitysympäristössä.

Pelimoottorista ei toimiteta valmiiksi käännettyä ohjelmatiedostoa vaan käyttäjän on tehtävä käännös (build) itse, mikä hidastaa AndEnginen käyttöönottoa. Esimerkkinä AndEnginellä toteutetusta Android-mobiilipelistä mainittakoon suosittu ilmaispeli Bunny Shooter.

Kosketusnäyttöohjausta AndEngine tukee melko hyvin. Moottorista löytyy valmiit funktiot kosketuksen havaitsemiselle ja kontrolloinnille. AndEngine sisältää myös valmiin BaseOnScreenControl-kirjaston, jonka avulla saadaan ruudulle tuotua virtuaalinen joystick-tyyppinen ohjain (kuva 6).

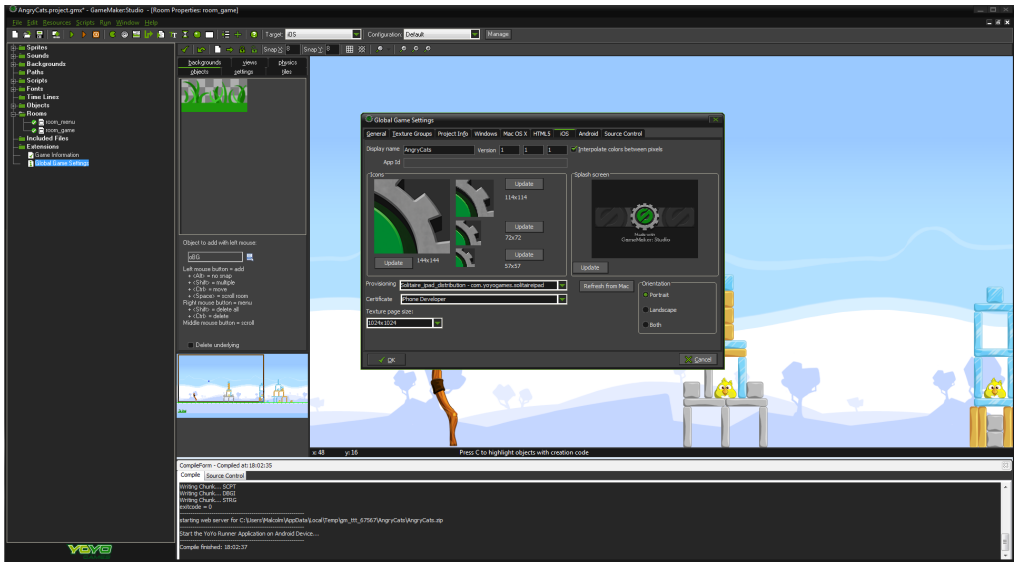


Kuva 6. AndEnginen virtuaalinen joystick-ohjain mobiililaitteen ruudulla.

BaseOnScreenControl-kirjaston ja virtuaalisen ohjaimen lisääminen vaatii pidemmän ohjelmakoodin kirjoittamista (liite 1). AndEnginen foorumeilta löytyy esimerkkejä tähän tarkoitukseen.

4.4 GameMaker

GameMaker on YoYo Gamesin kehittämä järjestelmäriippumaton 2D-pelimoottori. Järjestelmäriippumattomana se tukee monia käyttöjärjestelmiä: Windows, Windows Phone, Mac OS, iOS, Android ja Ubuntu. GameMakerin mukana saadaan oma kehitysympäristö GameMaker Studio. Ohjelmointikielenä käytetään GameMaker Studion omaa skriptikieltä. Moottorikohtainen kehitysympäristö (kuva 7) nopeuttaa sekä käyttöönottoa että pelinkehitystä huomattavasti. GameMakerin voi ladata ilmaiseksi osoitteesta http://www.yoyogames.com/studio.



Kuva 7. Ruutukaappaus GameMaker Studio -pelinkehitysympäristöstä.

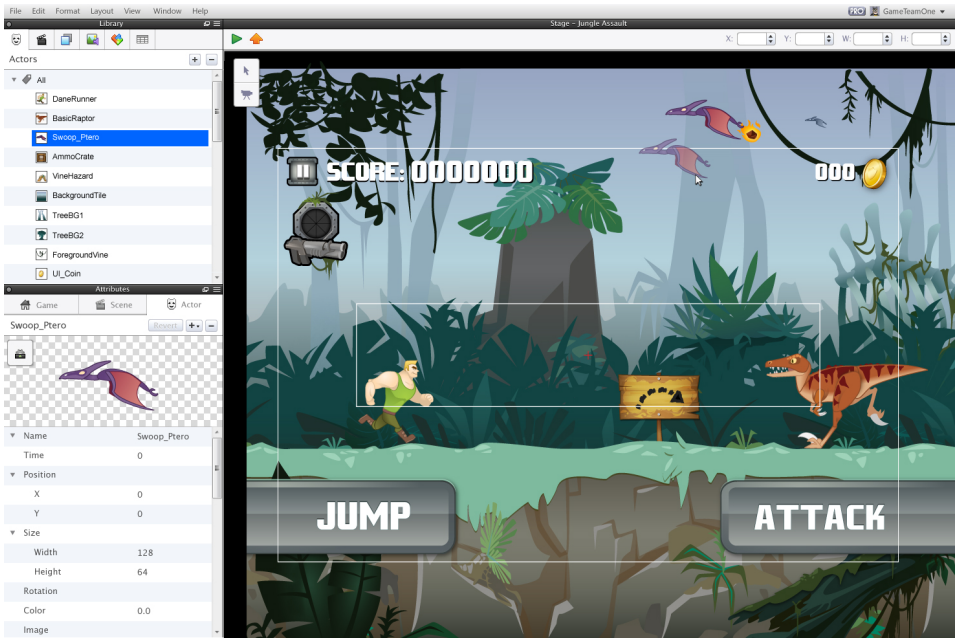
Ilmaisversiolla voi toteuttaa pelejä ainoastaan Windows-, WindowsPhone- ja Mac OS -laitteille. Android-laitteille lisenssi täytyy hankkia erikseen 199 Yhdysvaltain dollarilla. Asennuspaketti toimitetaan kätevästi yhtenä

itseasentuvana tiedostona. Esimerkkinä GameMaker Studiolla tehdystä pelistä voidaan mainita D-Pad Studion kehittämä Savant – Ascent. [28]

Kosketusnäyttöä GameMaker käsittelee hiiriohjauksena. Yksi napautus vastaa hiiren vasenta painiketta ja kaksoisnapautus hiiren oikeaa painiketta. Vastaavien funktioiden ”device_mouse_check_button”, ”device_mouse_x” ja ”device_mouse_y” kautta voidaan lukea napautuksen sijainti laitteen ruudulla. Eri laiteindeksit (device id) mahdollistavat viiden samanaikaisen kosketuksen havaitsemisen. [29] GameMakerista löytyy lisäksi tuki Android-laitteiden kiihtyvyysantureille ja värinäominaisudelle. Virtuaalista kosketusnäytöllä toimivaa joystick-ohjainta GameMakerin mukana ei tule, mutta sen totauttamiselle löytyy runsaasti esimerkkejä ohjelmiston tukifoorumeilta.

4.5 GameSalad

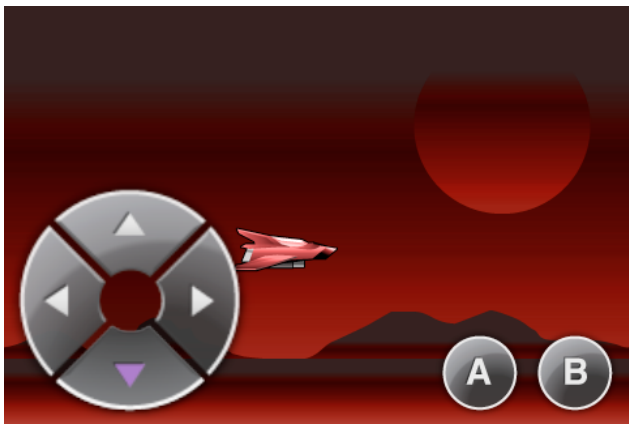
GameSalad Creator on samannimisen yrityksen, Game Salad Inc:n, toteuttama pelinkehitysympäristö. Ympäristö tarjoaa tuen laajalle valikoimalle päätelaitteita: Windows, iOS, Mac OS, Android sekä www-pienoisohjelmat (HTML5 widget). Alun perin GameSalad Creator oli saatavilla ainoastaan Mac-tietokoneisiin. Vuonna 2012 kehitysympäristöstä julkaistiin Windows-käännös, mikä mahdollisti myös Android-sovellusten kehittämisen. GameSalad Creator sisältää tehokkaan graafisen muokkaustyökalun (kuva 8), jonka avulla pelinkehitys onnistuu ilman aiempaa ohjelmointikokemusta. Yrityksen www-sivulla toimii lisäksi resurssikauppa, josta käyttäjät voivat ostaa grafiikka- tai äänitiedostoja peliinsä. GameSalad Creatorin perusversio on ilmainen ja ladattavissa osoitteesta http://gamesalad.com/.



Kuva 8. Ruutukaappaus GameSalad Creator -pelinkehitysympäristöstä.

Kehittyneemmät ominaisuudet vaativat pro-lisenssin, jonka hinta on 299 dollaria vuodessa. Android-laitteiden tuki kuuluu näihin kehittyneisiin ominaisuuksiin. Markkinoilta löytyy satoja tuhansia GameSalad Creatorilla kehitettyjä pelejä. Yhtenä esimerkkinä voidaan mainita suosittu pulmapeli The Secret of Grisly Manor. [30]

Kosketusnäyttöohjausta GameSalad tukee monikäyttöisen ja laiteriippumattoman ”Official Cross-Platform Controller Template”-valmispohjan kautta. Valmiiksi tehty kontrolliteema sisältää sekä analogisen ohjainmallin että ristiohjaimen (kuva 9).



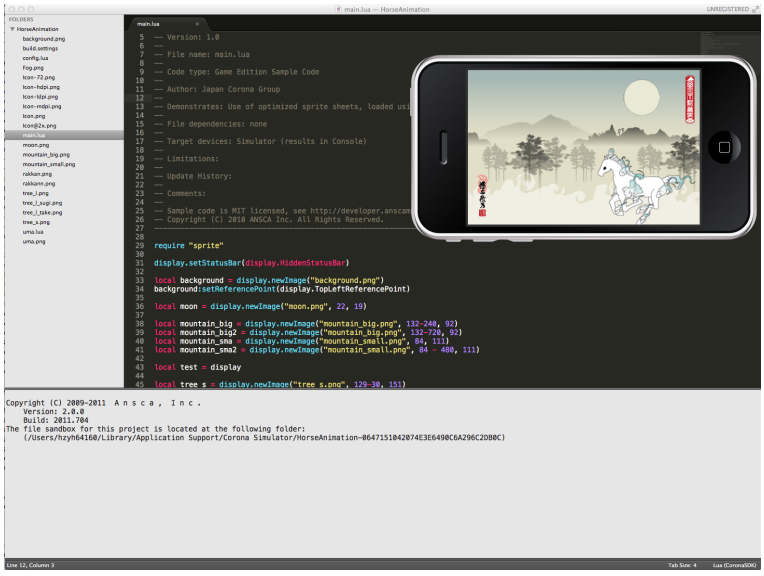
Kuva 9. Ristiohjaimen valmispohja GameSalad Creatorissa.

Graafisen kehitysympäristön ansiosta kosketusohjauksen toteuttamiseen tarvitaan hyvin vähän ohjelmointikokemusta. Kosketusnäyttösyötteiden lisäksi GameSalad kykenee lukemaan tietoa Android-laitteen kiihtyvyysantureilta.

Analogisen joystick-ohjaimen hyötyinä ovat liikkeen sulava jatkuvuus ja tarkkuus. Analogisena se tunnistaa suunnan lisäksi liikkeen voimakkuuden. Ristiohjaimen vahvuuksia ovat nopea reagointi ja helppokäyttöisyys. Se tunnistaa ainoastaan liikkeen suunnan. Liikkeen voimakkuus on ristiohjaimella aina samansuuruinen.

4.6 Corona

Corona SDK (Software Development Kit) on Corona Labsin kehittämä järjestelmäriippumaton 2D-pelinkehitysympäristö. Sen avulla voidaan kehittää pelisovelluksia niin iPhonelle, iPadille kuin Android-laitteillekin. Ohjelmointikielenä toimii C++/OpenGL-perustan päälle integroitu Lua-skriptikieli. Corona SDK sisältää laajan valikoiman kehitystyökaluja, joiden avulla voidaan hyödyntää kuvan ja äänen lisäksi mobiililaitteen verkko-ominaisuuksia, GPS-sijaintia, kosketusnäyttöä sekä kiihtyvyysantureita. Työkaluihin sisältyy simulaattori, Corona Simulator (kuva 10), jolla mobiilisovellusta voidaan testata ilman fyysistä mobiililaitetta.



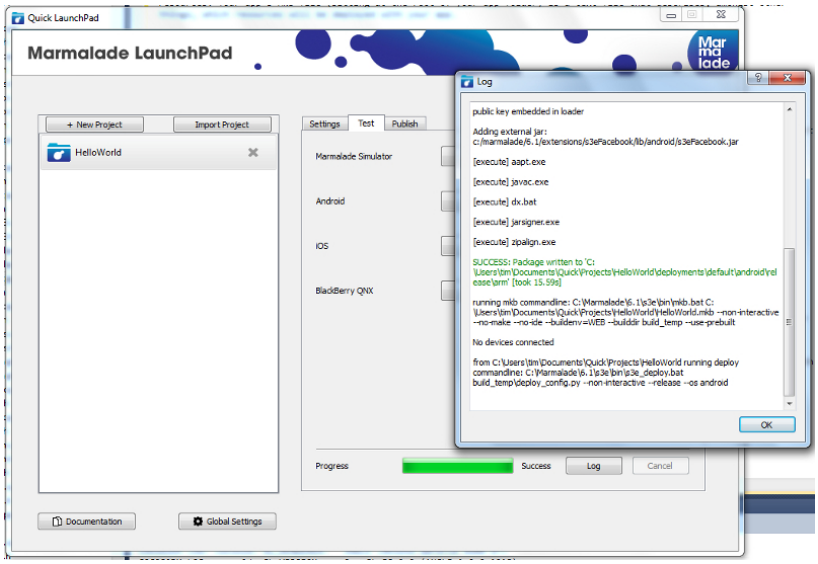
Kuva 10. Ruutukaappaus Corona SDK -kehitysympäristöstä.

Corona SDK:sta on saatavilla ilmainen Starter-versio sekä useita kattavammilla ominaisuuksilla varustettuja maksullisia versioita. Lataaminen vaatii rekisteröitymisen Corona Labsin www-sivulla osoitteessa http://coronalabs.com/products/corona-sdk/. Esimerkiksi mobiilipeli Freeze! on toteutettu käyttäen Corona SDK -työkaluja. [31]

Kosketusnäyttöohjausta Corona SDK tukee varsin kattavasti mukaan lukien napautukset, eleet sekä monikosketusominaisuudet. Valmiita ohjainmalleja pelinkehitysympäristö ei sisällä, mutta esimerkiksi joystick-moduulin ohjelmakoodi löytyy Coronan www-sivulta Code-osion alta. Kiihtyvyysanturin lukemia voidaan Coronassa iteroida accelerometer-tapahtuman kautta. [32]

4.7 Marmalade

Marmalade SDK on tehokkaaseen C++ -ohjelmointikieleen perustuva 3D-pelimoottori ja kehitysympäristö. Järjestelmäriippumattomana sillä voidaan kehittää sovelluksia sekä perinteiselle tietokoneelle että useille mobiililaitteille. Näihin mobiililaitteisiin lukeutuvat iOS, Android, BlackBerry, PlayBook OS, bada ja Windows Phone 8. Marmalade SDK sisältää simulaatio-ohjelman (kuva 11), jonka avulla kehitettävää peliä voidaan testata ilman fyysistä mobiililaitetta.



Kuva 11. Ruutukaappaus Marmalade SDK -kehitysympäristöstä.

Kokeiluversio tarjoaa ilmaisen 30 päivän tutustumisjakson Marmaladen kehitysympäristöön. Tämä on ladattavissa tuotteen sivuilta, osoitteesta https://developer.madewithmarmalade.com/. Pelin julkaiseminen vaatii kuitenkin Marmalade SDK -lisenssin hankkimisen. Android-lisenssien hinnat alkavat 149 dollarista vuodessa. Imperial Game Studion Golf Battle 3D on yksi esimerkki Marmaladella tuotetusta mobiilipelistä. [33]

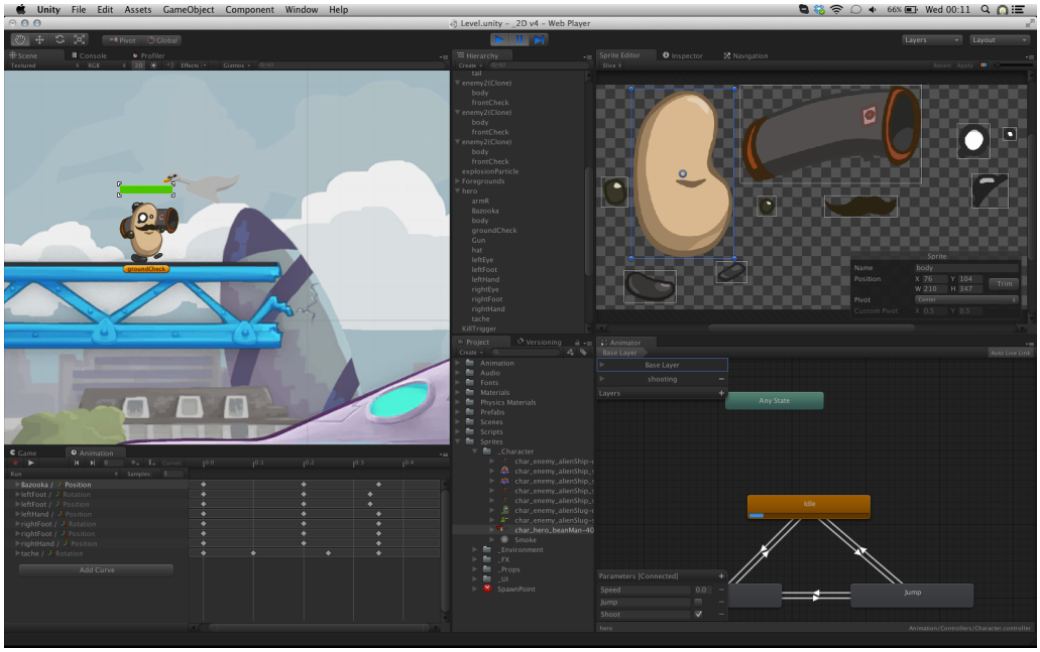
Marmalade kykenee lukemaan mobiililaitteiden syötteitä hyvin laaja-alaisesti. Moottori tukee napautuksia, pyyhkäisyjä sekä monikosketusta. Lisäksi laitteelta voidaan lukea virtuaalinäppäimistön komentoja ja kiihtyvyysantureiden lukemia. Ohjausteemoja ei toimiteta valmiina toiminnallisina elementteinä, vaan niiden toteutus on pelinkehittäjän itsensä vastuulla. [34]

4.8 Unity

Unity on Unity Technologiesin kehittämä 3D-pelimoottori ja pelinkehitysympäristö (kuva 12). Se on järjestelmäriippumaton ja sillä voidaan julkaista sovelluksia useille eri alustoille. Näihin alustoihin kuuluvat:

- iOS
- Android
- Windows Phone 8
- Microsoft Windows
- BlackBerry 10
- Mac OS
- Linux
- PlayStation 3
- Xbox 360
- Wii U
- Flash
- www-pienoisohjelmat.

Unityn www-sivulla toimivan Asset Storen kautta pelinkehittäjät voivat ostaa sovellukseensa valmiita komponentteja, kuten grafiikkaa, ääntä tai kokonaisia pelihahmoja. Pelimoottorista on saatavilla ilmainen versio, josta kuitenkin puuttuu useita kehittyneitä ominaisuuksia.



Kuva 12. Ruutukaappaus Unity-kehitysympäristöstä.

Kehittyneemmät ominaisuudet saadaan käyttöön ainoastaan hankkimalla ammattiversio eli Unity Pro -lisenssi. Tämän lisenssin hinta on 75 dollaria kuukaudessa tai 1500 dollaria kertamaksuna. Ilmaisversio on ladattavissa osoitteesta https://unity3d.com/. Unityllä on kehitetty valtava määrä mobiilipelejä; yhtenä esimerkkinä Rovion vuonna 2012 julkaisema pulmapeli Bad Piggies. [35]

Unityssä kosketusohjaus on toteutettu Input-rajapintaluokan avulla. Input.GetTouch-kutsulla voidaan lukea mobiililaitteelta yksittäisiä napautuksia tai pyyhkäisyjä. Monimutkaisempia eleitä ja monikosketusta luetaan Input.touches-taulukosta. Unityn mukana tulevasta ”Standard Assets (Mobile)” -paketista löytyy erilaisia valmiita ohjainasetelmia kosketusnäyttöjä varten.

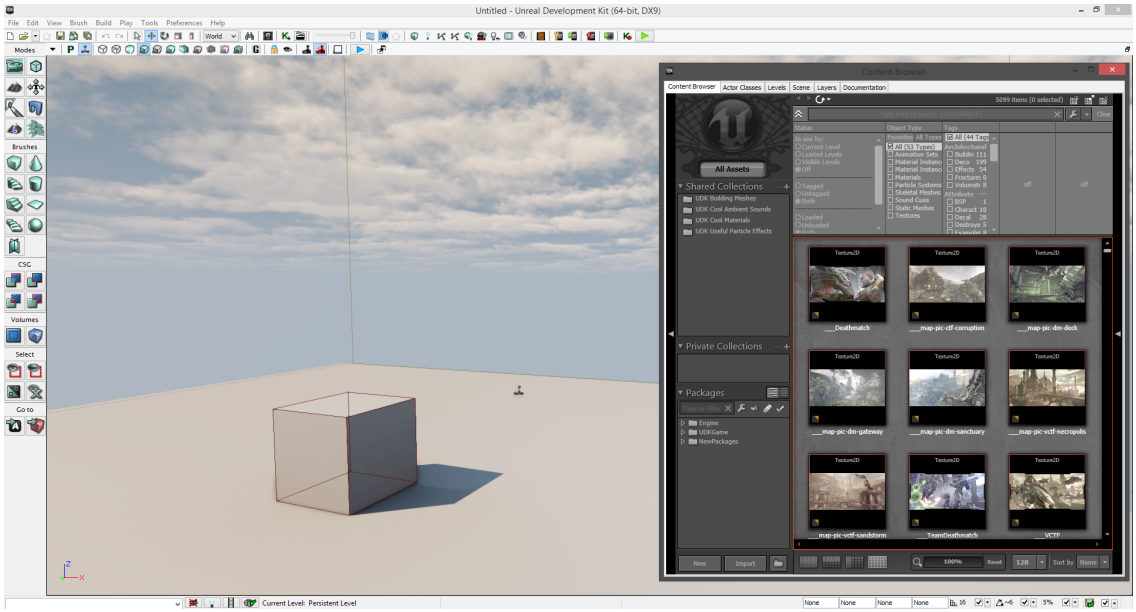
Kiihtyvyysantureiden ja GPS-sijainnin käyttö on myös mahdollista Input-luokan kautta. [36]

4.9 Unreal

UDK (Unreal Development Kit) on tunnettu pelinkehitysympäristö, joka perustuu Epic Gamesin Unreal Engine -pelimoottoriin. Pelimoottorin ensimmäinen versio julkaistiin jo vuonna 1998. Nykyinen versio on Unreal Engine 3, mutta neljättä versiota kehitetään aktiivisesti. Vahvasti 3D-puolelle painottuvaa pelimoottoria on käytetty paitsi tietokonepelien pohjana myös TV-mainoksissa ja erilaisissa simulaattoreissa. Useat PC- ja konsolipuolen hittipelit pohjautuvat Unreal Engineen sen graafisen näyttävyyden ja nopean ohjelmakoodin vuoksi. UDK:n avulla voidaan julkaista pelejä seuraaville alustoille:

- Microsoft Windows
- Xbox One
- Windows RT
- OS X
- Linux
- PlayStation 4
- iOS
- Android
- www-pienoissovellukset.

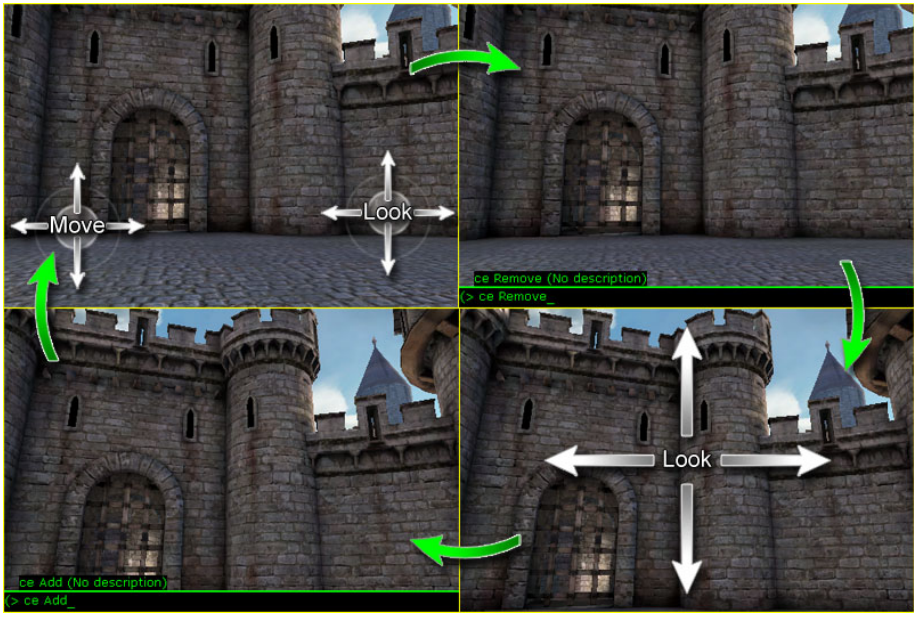
Unreal Development Kitistä (kuva 13) on saatavilla ilmainen versio osoitteessa https://www.unrealengine.com/products/udk. Ilmaisversio perustuu Unreal Enginen kolmanteen versioon ja sisältää kaikki tämän ominaisuudet.



Kuva 13. Ruutukaappaus Unreal Development Kit -kehitysympäristöstä.

Kaupallisten sovellusten julkaiseminen vaatii kuitenkin 99 dollarin lisenssimaksun sekä 25 % niistä tuotoista, jotka ylittävät 50 000 Yhdysvaltain dollaria. Lisenssimaksu tulee kuitenkin suorittaa vasta kun peli julkaistaan, joten kehitysvaiheen aikana Unreal Enginen ja sen kehitysympäristön käytöstä ei aiheudu kustannuksia. [37] Esimerkiksi Tower of Ascension -mobiilimoninpeli on toteutettu UDK:lla.

Unreal Development Kit hallitsee kosketusnäyttöohjausta kattavasti MobilePlayerInput-luokan kautta. Luokasta löytyvät komennot napautusten ja eleiden lisäksi kiihtyvyysantureiden lukemiseen. UDK:n syötteenhallintatyökalulla Kismetillä voidaan vaivattomasti luoda uusia ohjausasetelmia ja vaihtaa niiden välillä dynaamisesti pelin aikana (kuva 14). [38]



Kuva 14. Kismet ja ohjausasetelman muuttaminen dynaamisesti [38].

Ohjainasetelman muuttaminen kesken pelin voi olla tarpeellista, jos pelihahmon toiminta muuttuu olennaisesti. Tällaisia tilanteita voivat olla esimerkiksi inventaarion avaaminen, jonkin kulkuneuvon käyttöönotto tai elokuvamainen välikohtaus. Tarpeetonta ohjainasetelman vaihtamista on kuitenkin syytä välttää, sillä uuden ohjaustavan opettelu voi häiritä pelaajaa ja sekoittaa pelikokemusta.

4.10 Lopputulos

Vertailun päätteeksi pelimoottorit järjestettiin listaksi sopivuuden mukaan. Järjestyksessä otettiin huomioon moottoreiden ominaisuudet (taulukko 1), projektin tavoitteet sekä käytettävissä olevat resurssit. Lopullinen lista pelimoottoreiden sopivuudesta oli muotoa:

1. Unity
2. Unreal
3. GameSalad

4. GameMaker
5. Corona
6. Marmalade
7. AndEngine.

Pelimoottorivertailun perusteella päätettiin PenQ Rocketin pelimoottoriksi valita Unity. Sen etuina ovat ilmainen Android-tuki, helppo käytettävyys sekä startup-yritykselle sopiva hinnoittelu. Lisäksi Unityn pelimoottori tukee hyvin 2D-pelejä, ja sen mukana toimitetaan tehokas graafinen kehitysympäristö.

Taulukko 1. Pelimoottoreiden ja niiden kehitysympäristöjen ominaisuudet.

Ominaisuus	Unity	Unreal	Game-Salad	Game-Maker	Corona	Marmalade	AndEngine
Android-tuki	X	X	X	X	X	X	X
2D-tuki	X		X	X	X	X	X
3D-tuki	X	X				X	
Graafinen SDK	X	X	X	X			
Helppokäyttöinen	X		X	X	X		
Tunnettuja pelejä	X	X	X		X	X	X
Hyvät tukipalvelut	X	X	X		X		
Laajennettavuus	X	X	X	X		X	
Minimikustannus	0 $	99 $	299 $	199 $	0 $	149 $	0 $

Unityn tukipalvelut koettiin hyödyllisiksi. Verkkosivuilta löytyi runsaasti ohjaavia videotutoriaaleja ja ilmaiseksi ladattavia testiprojekteja. Unityn ympärille on kokoontunut aktiivinen kehittäjäyhteisö, mikä kielii elinvoimaisesta tuotteesta. Tällaiselle tuotteelle voidaan tulevaisuudessakin odottaa uusia päivityksiä ja pikaista ongelmien korjaamista.

5 KOSKETUSNÄYTTÖOHJAUS UNITYSSÄ

Tässä luvussa käsitellään kosketusnäyttöohjauksen toteuttamista Unityn kehitysympäristössä. Tekstissä käydään läpi tärkeimmät kosketusohjaukseen liittyvät metodit ja niiden toiminta.

5.1 Input Manager

Unityn sisään rakennettua peliohjausta voidaan määrittää Input Managerin kautta. Input Manager saadaan auki valitsemalla päävalikosta ”Edit > Project Settings > Input”. Tällöin Inspector-ikkunaan aukeaa ohjainmäärittelytaulukko (kuva 15).

Kuva 15. Unityn Input Manager.

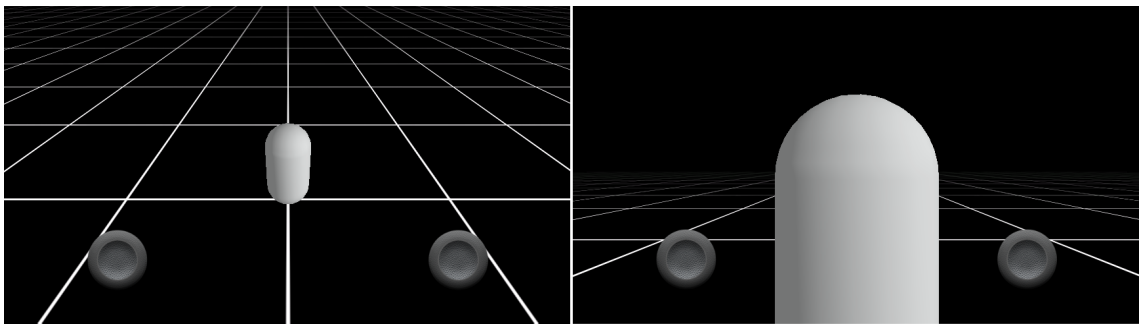
Kosketusnäyttöohjausta ei kuitenkaan voida suoraan määrittää Input Managerista. Tätä varten on projektiin tuotava Standard Assets (Mobile) -paketti tai vaihtoehtoisesti kommunikoitava ohjelmakoodissa suoraa Input-luokan kanssa.

5.2 Standard Assets (Mobile) -paketti

Standard Assets (Mobile) -paketti sisältää mobiiliohjauksen kannalta tärkeät elementit. Se voidaan liittää mihin tahansa projektiin valitsemalla päävalikosta ”Assets > Import Package... > Standard Assets (Mobile)”. Paketti sisältää kuusi valmista ohjainasetelmaa:

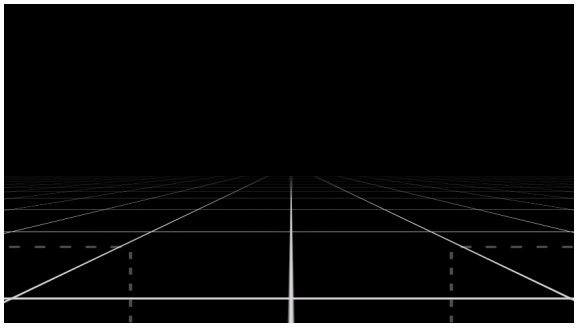
- Camera Relative Setup
- First Person Setup
- First Person Tilt
- Player Relative Setup
- Sidescroll Setup
- Tap Control Setup.

Camera Relative Setup ja Player Relative Setup -ohjainasetelmat (kuva 16) ovat varsin samanlaisia ja perustuvat kahteen virtuaaliseen joystick-ohjaimeen. Camera Relative Setupissa liikutaan kameran suhteen ja Player Relative Setupissa pelaajahahmon suhteen.



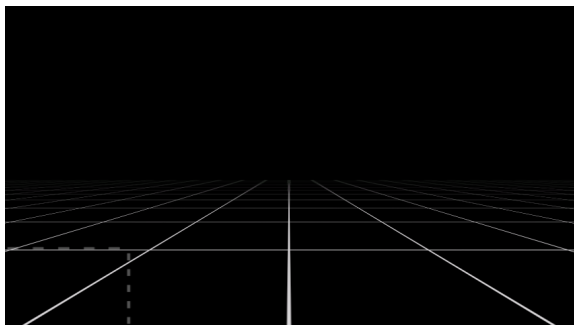
Kuva 16. Camera Relative Setup ja Player Relative Setup -ohjainasetelmat.

First Person Setup mallintaa tilannetta, jossa katsotaan pelaajahahmon silmistä (kuva 17). Joystick-ohjaimia ei tässä asetelmassa ole, mutta ruudun alakulmat toimivat kosketusalueina.



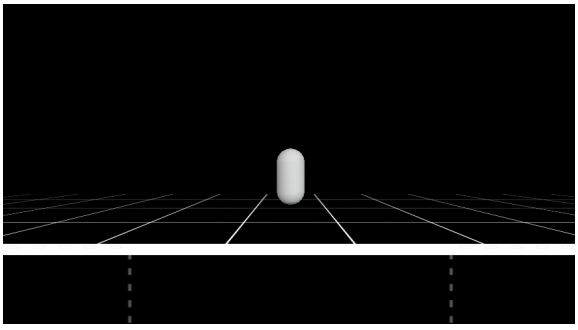
Kuva 17. First Person Setup -ohjainasetelma.

First Person Tilt -ohjainasetelma (kuva 18) hyödyntää mobiililaitteen kiihtyvyyssensoreita kamerasuunnan määrittämiseksi. Vasen alakulma toimii kosketusalueena, jonka avulla pelaaja voi liikkua.



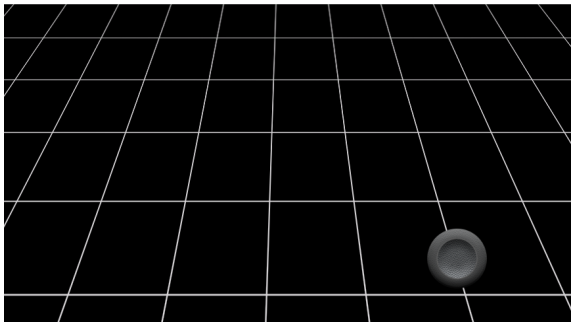
Kuva 18. First Person Tilt -ohjainasetelma.

Sidescroll Setup (kuva 19) puolestaan mallintaa sivusta nähtyä tasohyppelypelin kontrollimekaniikkaa. Kameraa ei voi tässä asetelmassa kääntää, sillä kuvakulma on lukittu. Ruudun alakulmat toimivat kosketusalueina, joiden avulla voidaan liikkua eteen- ja taaksepäin.



Kuva 19. Sidescroll Setup -ohjainasetelma.

Tap Control Setupissa (kuva 20) liikutaan napauttamalla minne tahansa ruudulla. Tässä asetelmassa kamera kuvaa oletuksena hieman yläviistosta. Ohjainasetelma sisältää myös joystickin kameran kääntämiseen.



Kuva 20. Tap Control Setup -ohjainasetelma.

Standard Assets (Mobile) -paketti sisältää lisäksi ohjelmakoodit kaikkiin edellisiin ohjainasetelmiin, jotta pelinkehittäjät voivat vapaasti muokata niitä omien tarpeidensa mukaan.

5.3 Input-luokka

Unity tarjoaa mahdollisuuden myös niille, jotka haluavat toteuttaa oman ohjainasetelman. Input-luokka on Unityn rajapinta kaikkeen käyttäjäsyötteeseen. Tämän luokan kautta voidaan lukea ruudulla havaittavia kosketuksia joko Input.GetTouch-metodin avulla tai suoraan Input.touches -taulukosta. Kummatkin edellisistä palauttavat olioita, jotka ovat Touch-luokan ilmentymiä. [36]

5.4 Touch-luokka

Touch-olioluokka mallintaa ruudulla havaittua kosketusta. Nämä oliot pitävät sisällään kaiken kosketuksesta havaitun tiedon. [36] Luokasta voidaan lukea kuusi muuttujaa:

- deltaPosition

- deltaTime
- fingerId
- phase
- position
- tapCount.

DeltaPosition-muuttuja ilmoittaa kosketuksen sijainnin muutoksen edelliseen mittaustulokseen verrattuna. Sijainti annetaan Vector2-muodossa eli se sisältää sekä x- että y-koordinaatin muutoksen. [36] Tästä muutoksesta voidaan laskea kosketuksen liikesuunta ja liikkeen nopeus.

DeltaTime-muuttuja kertoo kuinka kauan aikaa on kulunut edellisestä kosketusmittauksesta. [36] Aika voi vaihdella hiukan, vaikka Unity pyrkii mittaamaan kosketusta tasaisin väliajoin. Tätä muuttujaa voidaan hyödyntää, kun halutaan laskea kosketuksen liikenopeus hyvin tarkasti.

FingerId-muuttujasta voidaan lukea kyseisen kosketuksen indeksinumero. Mikäli ruudulla havaitaan useampia kosketuksia yhtä aikaa, erotellaan kosketukset toisistaan tämän indeksinumeron avulla. [36]

Phase-muuttujan arvosta nähdään kosketuksen vaihe. Erilaisia mahdollisia vaiheita on viisi: Began (alkoi), Moved (liikkui), Stationary (pysyi paikallaan), Ended (päättyi) ja Canceled (peruuntui). [36] Näiden avulla voidaan tarkkailla kosketuksen tilaa ilman monimutkaisia numeerisia laskuja. Esimerkiksi kosketuksen päättymiseen voidaan lisätä ääniefekti tarkkailemalla sen vaihetta.

Position-muuttuja kertoo kosketuksen sijainnin ruudulla. Sen arvot ovat Vector2-muotoa eli ne sisältävät sekä kosketuksen x- että y-koordinaatin. [36] Näiden avulla on mahdollista toteuttaa erilaisia efektejä käyttäjän koskettaessa näyttöruudun eri osia. On hyvä huomata, että objektien napauttaminen voidaan myös toteuttaa OnMouseDown-funktion kautta, sillä napautukset toimivat myös hiiren painikkeina. Tämä on usein helpompi toteutustapa kuin kosketuksen sijainnin laskeminen ja vertaaminen peliobjektin sijaintiin.

TapCount-muuttujan arvo näyttää kokonaislukuna kuinka monta kertaa samalle ruutualueelle on napautettu peräkkäin. Tätä käytetään tyypillisesti kaksoisnapautuksen tunnistamiseen. Kaikki Android-laitteet eivät lue perättäisten napautusten lukumäärää eivätkä siten tue tätä muuttujaa. [36]

6 PENQ ROCKET

Opinnäytetyön käytännön osana toteutettiin kosketusohjattavan Android-mobiilipelin prototyyppi. PenQ Rocket on tyylitelty kaksiulotteinen ongintaseikkailu etelänavan miljöössä. Pelissä liikutaan karttanäkymän avulla eri apajille, joilla kullakin on omat saalistyyppinsä. Kalansaaliit pelaaja myy pingviinikylässä ja saa vaihtokaupassa ongintaa helpottavia esineitä.

6.1 Työkalut ja kehitysympäristö

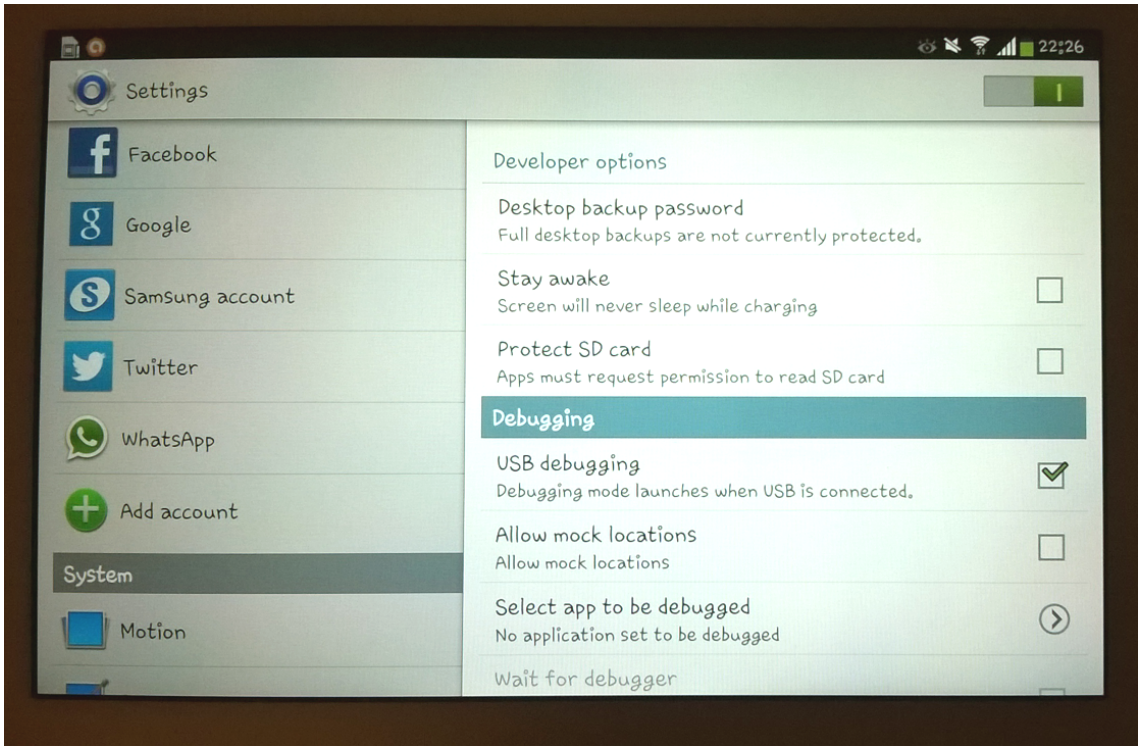
Ennen mobiilipelin toteutusvaihetta hankittiin ja asennettiin tarvittavat ohjemistokehitystyökalut, joilla Android-mobiilipeliä voitiin kehittää. Tämä onnistui kuuden askeleen kautta:

1. Unityn asennuspaketin lataaminen ja asentaminen
2. Android SDK ADT Bundle -asennuspaketin lataaminen ja asentaminen
3. mobiililaitteen ajurien lataaminen ja asentaminen
4. Android-laitteen kytkeminen testaustilaan (debug-tila)
5. Unityn yhdistäminen Android SDK:n kanssa
6. Android-laitteen kytkeminen tietokoneeseen USB-kaapelilla.

Pelimoottoriksi oli valittu Unity, joka ladattiin ilmaiseksi osoitteesta https://unity3d.com/. Seuraavaksi tarvittiin Android-työkalupaketti, Android SDK. Android SDK on niin ikään ilmaiseksi jaettava työkalukokoelma, joka ladattiin osoitteesta http://developer.android.com/sdk/. Tietokoneen ja mobiililaitteen välisen kommunikaation mahdollistamiseksi asennettiin tietokoneeseen kyseisen mobiililaitteen ajurit. Nämä etsittiin ja ladattiin mobiililaitevalmistajan kotisivuilta.

Android-laite tuli kytkeä testaustilaan, jotta se salli tietokoneen lähettää tiedostoja laitteen muistiin. Testaustilaan päästiin vanhemmissa Android-

laitteissa valikoiden kautta valitsemalla ”Settings > Developer options”. Alkaen versiosta 4.2 (Jelly Bean) tämä valikko on piilotettu, ja sen näyttämiseksi tuli valita ”Settings > About Phone > Build Version”. Build Version -tekstiä tuli tämän jälkeen napauttaa useita kertoja, jolloin turvallisuussyistä piilotettu Developer options -valikko tuli näkyviin. Lopuksi vaihdettiin USB debugging mode ON-asentoon (kuva 21).

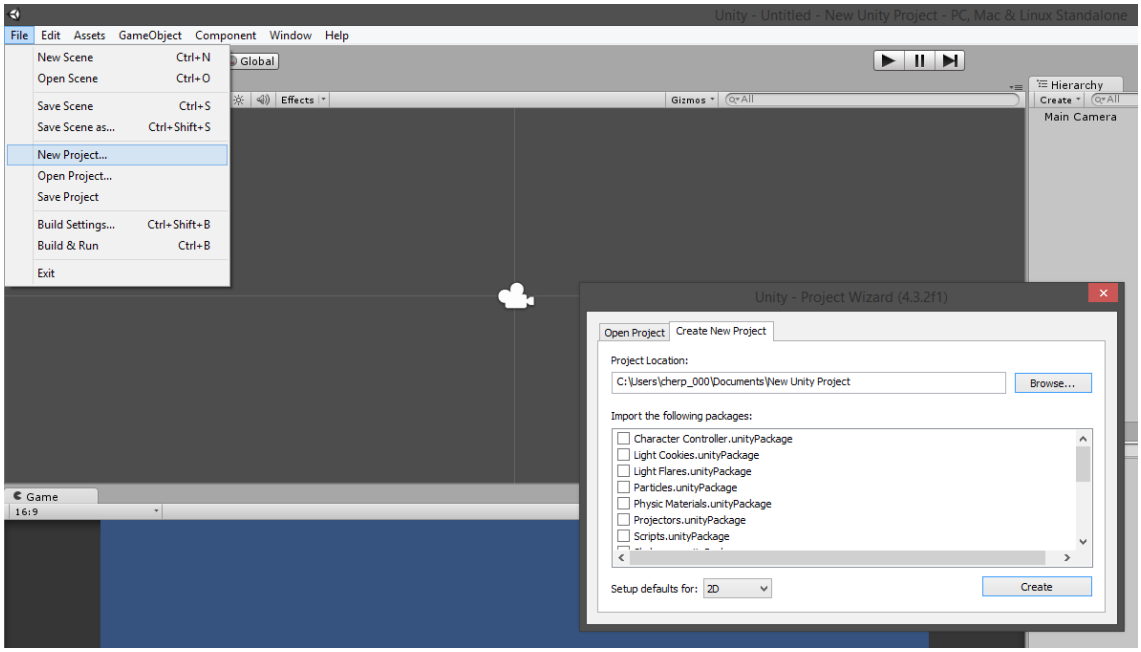


Kuva 21. USB-testaustilan salliminen Android-tabletilla.

Unityn yhdistäminen Android SDK:iin hoidettiin Unityn valikosta. Valittiin ”Unity > Preferences > External tools”. Tämän jälkeen Android SDK:n asennushakemiston osoite kirjoitettiin sille varattuun kenttään. Lopuksi Android-laite kytkettiin tietokoneeseen laitteen mukana tulleen USB-kaapelin välityksellä.

6.2 Projektin luominen Unityssä

Unity käynnistettiin, tervetuloikkuna suljettiin ja perusvalikosta valittiin ”Unity > New Project...”. Auenneesta ikkunasta (kuva 22) valittiin projektille sopiva sijainti tietokoneen massamuistissa. Lisäksi ”Set defaults for”-pudotusvalikosta valittiin ”2D”, koska oltiin luomassa uutta kaksiulotteista peliä. ”Import the following packages”-kohdasta olisi ollut mahdollisuus valita projektiin lisäominaisuuksia, mutta niille ei toistaiseksi koettu tarvetta.



Kuva 22. Uuden projektin luominen Unityssä.

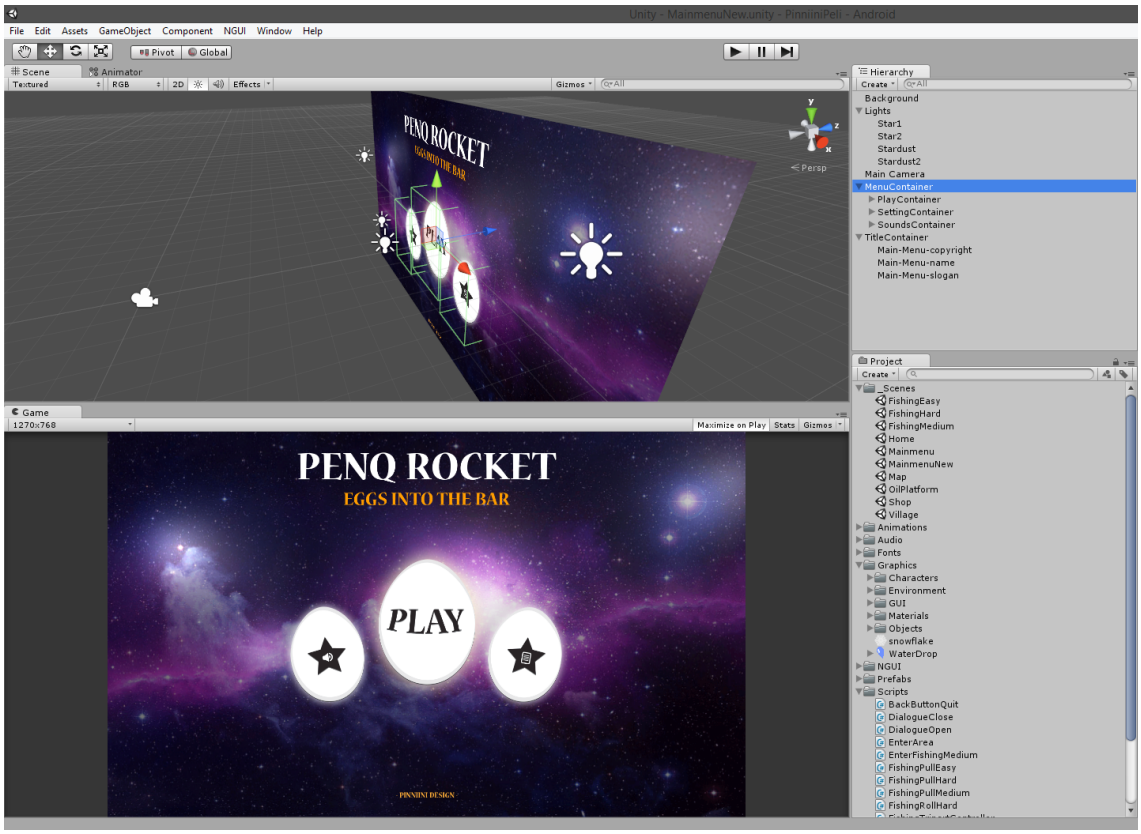
Lopuksi valittiin ”Create”, jolloin Unity loi projektille tarvittavat perustiedostot. Tämän jälkeen kehitysympäristö oli valmis uuden projektin kehittämiselle.

Organisointisyistä projektin Assets-hakemiston alle luotiin vielä omat alihakemistot grafiikalle, äänille, ohjelmakoodille sekä animaatioille. Lisäksi sovittiin standardi tapa tiedostojen nimeämiselle. Nimeämistavaksi valittiin ohjelmointipuolelta tuttu CamelCase-notaatio, jossa sanat kirjoitetaan isoin alkukirjaimin ja ilman välilyöntejä. Yhtenäisellä nimeämisellä ja rakenteella helpotettiin tiedostojen löytämistä, mikä osaltaan nopeutti työntekoa.

Ohjelmointikieleksi valittiin C# (C sharp). Sen tiedettiin olevan varsin nopea kieli käännettynä ja toimivan hyvin Unityn kanssa. Lisäksi C# oli ohjelmointiryhmälle ennestään tuttu. Projektin yhteisellä ohjelmointikielellä saavutettiin ohjelmakoodin kopioitavuus ja uudellenkäytettävyys.

6.2.1 Päävalikko

Päävalikon kautta pelaajan oli helppo siirtyä PenQ Rocketin pelimaailmaan. Valikosta voitiin aloittaa uusi peli, säätää äänenvoimakkuutta ja muita peliominaisuuksia. Päävalikkonäkymä toteutettiin omana Unity-tapahtumapaikkanaan (scene). Näkymä rakennettiin viidestä osakokonaisuudesta: taustasta, valonlähteistä, otsikkoteksteistä, valikkopainikkeista ja kamerasta (kuva 23).



Kuva 23. Päävalikon rakenne Unityssä.

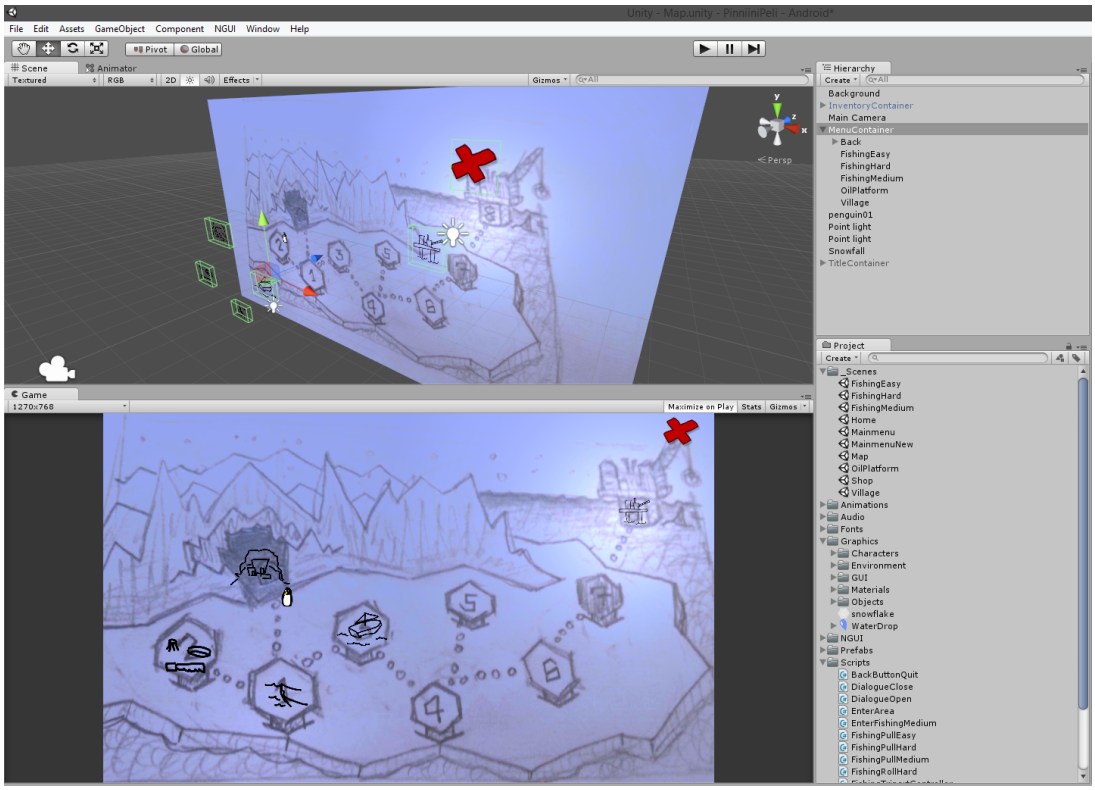
Osakokonaisuuksien myötä päävalikkoon saatiin yksinkertainen mutta toimiva hierarkia, mikä helpotti kunkin osan käsittelyä. Visuaalisen näyttävyyden lisäämiseksi valonlähteisiin ja valikkopainikkeisiin sijoitettiin animaatiokomponentit. Valikkopainikkeisiin kirjoitettiin lyhyet C# -ohjelmakoodit, jotka vastaanottavat kosketusnäytöltä saatavia komentoja.

Kosketusnäyttöohjaus otettiin päävalikossa huomioon sekä painikkeiden sijoittelussa että koon valinnassa. Yleisten suositusten mukaan kosketusnäyttöpainikkeiden tulisi olla ruudulla vähintään 22 x 22 mm kokoisia. Näytön pikselitiheydestä riippuen painikkeen vähimmäiskoko pikseleinä siis vaihtelee. Esimerkiksi testattavalla Samsung Galaxy S3 -laitteella pikselitiheys oli 306 dpi. Yhdellä senttimetrillä näyttöpisteitä oli siis 306 px / 25,4 mm eli noin 12 px / mm. Täten painikkeen suositeltu minimikoko kyseisellä laitteella oli 22 mm * 12 px / mm, eli 264 px. Päävalikon painikkeiden koot pyrittiin pitämään tämän minimin yläpuolella. [39]

6.2.2 Karttanäkymä

Karttanäkymän tehtävänä oli yhdistää pelin eri tapahtumapaikat ja tarjota pelaajalle yleiskuva PenQ Rocket -maailmasta. Näkymä toteutettiin neljällä osakokonaisuudella: taustalla, valonlähteillä, painikkeilla sekä kameralla.

Taustan ja valonlähteiden avulla luotiin pelaajalle mielikuva arktisesta maailmasta (kuva 24). Karttanäkymän painikkeet piilotettiin kuvien muotoon, sillä ne edustivat arktisen maailman eri lokaatioita. Ne ohjelmoitiin toimimaan kuten päävalikon painikkeet. Jokaiselle painikkeelle tuli erikseen määrittää kohteeksi tarkoitettu tapahtumapaikka, johon peli siirtyisi painikkeen aktivoituessa. Tapahtumapaikkasiirtymien suuresta määrästä johtuen luotiin yksi kompakti ohjelmakoodi (liite 2), jolle syötettiin parametrina siirtymän haluttu määränpää. Nyt samaa ohjelmakoodia voitiin käyttää jokaisessa siirtymäpainikkeessa. Määränpää määriteltiin julkiseksi (public) muuttujaksi, joten sitä voitiin muuttaa Unityn graafisella käyttöliittymällä.

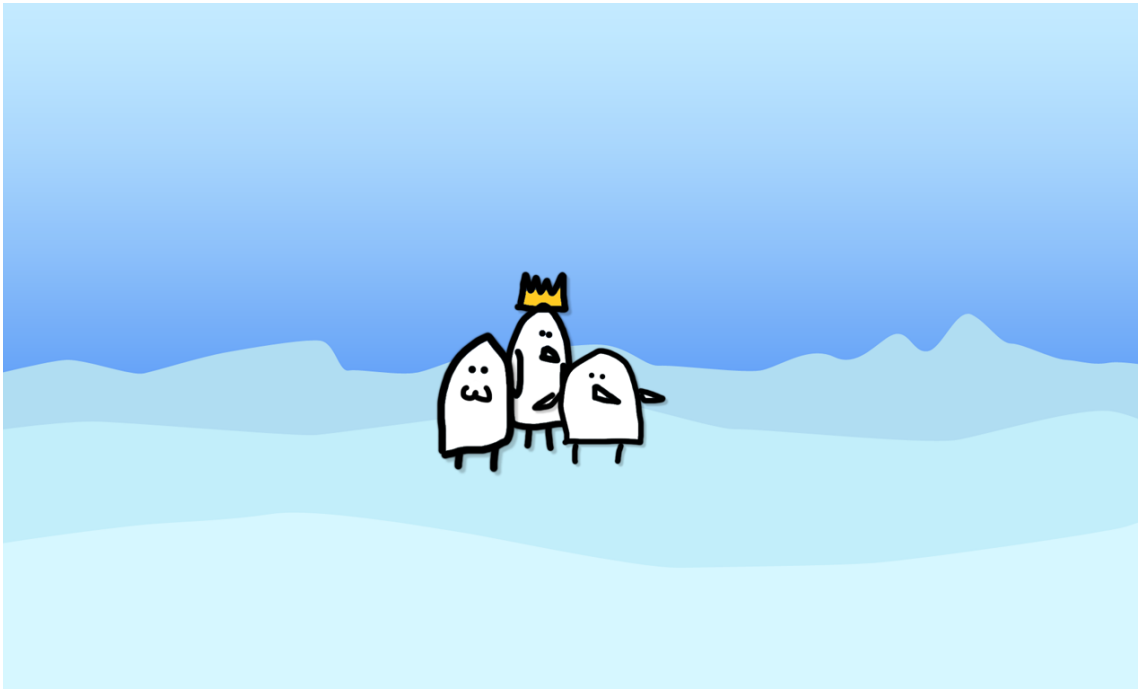


Kuva 24. Karttanäkymän muokkaus Unityssä.

Karttanäkymässä kosketusnäyttöohjaus otettiin huomioon etenkin painikkeiden sijoittelussa. Ongintapisteet ja pingviinikylä sijoitettiin etäälle toisistaan, jotta virhepainallukset saatiin minimoitua. Etäisyyksissä noudatettiin 1 cm minimietäisyyttä, joka vastasi testilaitteen ruudulla 120 pikseliä. Siltä varalta että kaikki kohteet eivät mahtuisi ruutuun, suunniteltiin käytettäväksi ruudunvieritystä kahden sormen kosketuseleellä. Tätä ei kuitenkaan toteutettu prototyypissä, sillä ruututila riitti kaikille kohteille.

6.2.3 Kylänäkymä

Kylänäkymän tarkoituksena oli toimia kohtauspaikkana pelaajan hahmon ja muiden arktisen maailman olentojen välillä (kuva 25). Kylässä pelaaja pystyi käymään kauppaa, keskustelemaan toisten pingviinien kanssa ja hankkimaan tehtäviä.

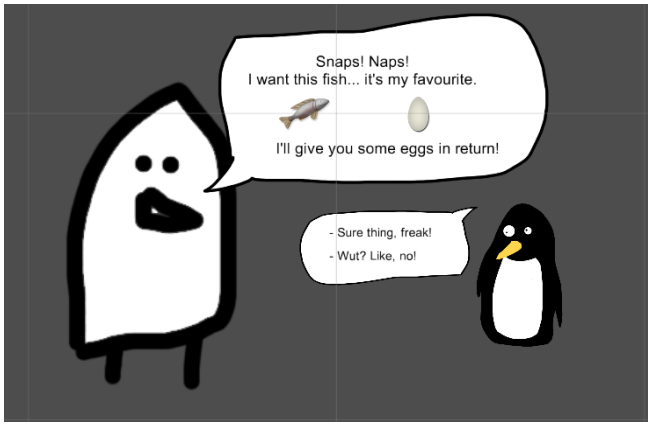


Kuva 25. Alustava kylänäkymä ja kylän muita pingviinejä.

Kylänäkymän rakenne toteutettiin edellisistä tapahtumapaikoista opitulla tavalla. Tällä kerralla valikkopainikkeiksi lisättiin kylän asukkaita sekä rakennuksia. Kullekin painikkeelle luotiin oma C# -ohjelmakoodi, koska henkilöiden ja rakennusten reaktiot pelaajan lähestyessä erosivat toisistaan huomattavasti. Niihin painikkeisiin, jotka edustivat pingviinihahmoja, kytkettiin dialogi-ikkunan avaaminen.

6.2.4 Dialogi

Dialogin tarkoituksena oli kuljettaa pelin juonta eteenpäin ja syventää pelaajan tunnesidettä peliin. Dialogit haluttiin pitää lyhyinä ja huvittavina, jotta ne eivät katkaisisi pelikokemusta. Näkymä toteutettiin leijuvana ikkunana, jonka takaa muu peliympäristö edelleen näkyi (kuva 26).



Kuva 26. Alustava dialogi-ikkuna ilman taustaa.

Rakenteellisiksi osakokonaisuuksiksi valittiin keskustelevat hahmot ja puhekuplat sisältöineen. Puhekupliin lisättiin dynaamiset tekstikentät, joiden avulla tekstisisältöä voitiin muuttaa ohjelmakoodikomennoilla. Pelaajahahmon puhekupliin toteutettiin myös kosketusohjattavat painikkeet eri vastausvaihtoehdoille. Kameraa tai taustaa ei tässä tarvittu, sillä ne olivat jo valmiina takana näkyvässä tapahtumapaikassa.

6.2.5 Varustekauppa

Varustekauppa tarjosi pelaajalle mahdollisuuden hankkia parempia kalastusvälineitä ja siten kehittyä kalastajana ja edetä pelissä. Kauppa toteutettiin dialogi-ikkunan tapaan. Varusteiden ostamista varten kirjoitettiin monimutkaisempi ohjelmarakenne, joka kysyy pelaajalta mistä varusteesta hän oli kiinnostunut. Seuraavaksi tarkistettiin oliko pelaajalla tarpeeksi resursseja varusteen hankkimiseksi, ja lopulta lisättiin valittu varuste pelaajan inventaarioon.

6.2.6 Inventaario

Inventaario eli pelaajahahmon mukanaan kantama varustus toteutettiin dialogin tavoin leijuvana ikkunana. Seuraavaksi ikkunaan lisättiin taustagrafiikka, jotta se

erottuisi paremmin pelimaailmasta. Inventaarion sisältö muodostettiin eri esineitä edustavista kuvista sekä numeroindikaattoreista, jotka ilmaisivat kunkin esineen määrän. Sisältö saatiin vastaamaan pelaajan omistamien esineiden määrää ohjelmakoodin (liite 3) avulla. Aina kun inventaarioikkuna avattiin, ohjelma asetti inventaarion esinemäärät oikeiksi.

Inventaario haluttiin mukaan useaan tapahtumapaikkaan, joten siitä tehtiin valmiselementti (prefab). Tämä tapahtui raahaamalla haluttu objekti tapahtumapaikan hierarkiasta koko projektin kattavan hierarkian alle. Inventaarion nimi muuttui väriltään siniseksi, mikä merkitsi objektin olevan valmiselementin ilmentymä (instance). Valmiselementin hyötynä oli uudelleenkäytettävyys. Samaa elementtiä voitiin käyttää useammassa paikassa, ja se toimi aina samalla tavoin. Kun inventaarioon haluttiin myöhemmin tehdä muutoksia, se onnistui helposti valmiselementtiä muuttamalla. Elementtiin tehdyt muutokset heijastuivat automaattisesti kaikkialle, missä tämän ilmentymiä oli käytetty.

6.2.7 Kalastusnäkymä

Kalastusnäkymä toimi pelin ytimenä, jossa varsinainen pelaaminen tapahtui. Tässä näkymässä pelaaja hankki lisää resursseja eli erilaisia kaloja ja muita löytöjä, joita merenpohjasta saattoi löytää. Saaliin nappaamiseksi ongella, pelaajan tuli suorittaa tarkkuutta tai nopeutta vaativa ohjaustoimenpide kosketusnäytöllä. Toimenpiteen helpottamiseksi ruudulle piirrettiin ongintaa kuvastava rengaselementti (kuva 27).

Kuva 27. Kalastusnäkymä sekä ongintaa kuvaava rengaselementti.

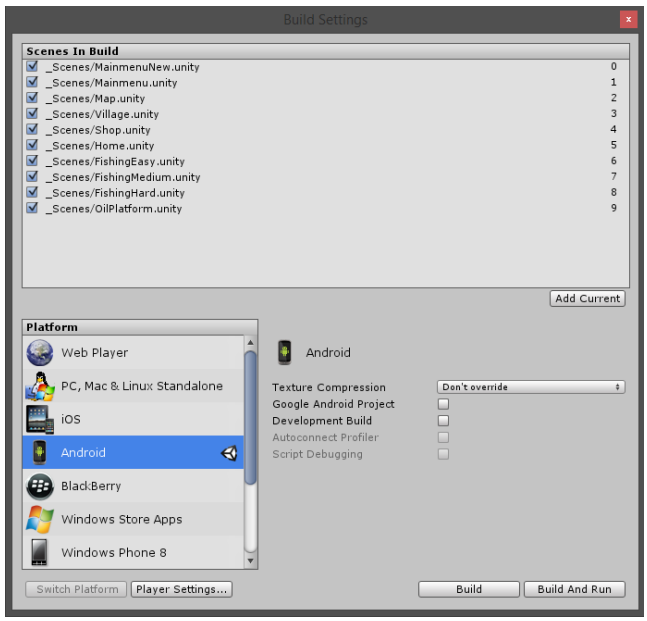
Kalastusnäkymä oli pelin dynaamisin osa. Ohjelmakoodissa tarkkailtiin käyttäjältä tulevia komentoja, joiden perusteella ohjattiin pelihahmon ja -ympäristön käyttäytymistä. Toteutuksen monimutkaisuutta lisäsi kahden ylimääräisen onginta-alueen lisääminen, jolla kullakin oli erilainen vaikeustaso.

6.3 Pelin tuominen mobiilialustalle

Unity-kehitysympäristössä peliä testattiin helposti ja nopeasti tietokoneruudulla. Kosketusnäyttöohjauksen kokeilemiseksi PenQ Rocket pyrittiin kuitenkin tuomaan Android-laitteelle jo kehityksen alkuvaiheessa. Mobiililaitteella testaamista jatkettiin koko kehitysprosessin ajan.

PenQ Rocketin tuominen päätelaitteelle tehtiin seuraavasti. Unityn valikosta valittiin ”File > Build settings...”. Build settings -ikkunasta haettiin Platform-kohdasta päätealustaksi Android ja painettiin Switch Platform -painiketta (kuva 28). Näin Unity pyrkii käännettäessä siirtämään valmiin ohjelman suoraan

Android-laitteeseen. Laite kytkettiin Unityä ajavaan tietokoneeseen USB-kaapelilla. Lopuksi valittiin ”File > Build & Run...”.



Kuva 28. Unity-projektin kääntäminen Android-laitteelle.

Unityn ruudulle ilmestyi edistymispalkki, joka ilmoitti kääntämisen etenemisvaiheen. Käännös kesti keskimäärin noin 10 s, minkä jälkeen PenQ Rocket avautui Android-laitteeseen.

6.4 Palaute ja jatkokehitys

PenQ Rocketin kehitystä jatketaan opinnäytetyön jälkeen. Projektin aikana suoritettiin useita pienimuotoisia käyttäjätestauksia. Näistä saadun alustavan palautteen myötä pelattavuutta on tarkoitus kehittää edelleen intuitiivisempaan suuntaan pelin seuraavassa versiossa. Pelaajalle annettavan informaation määrää lisätään, jolloin hänellä on parempi käsitys siitä, mitä kussakin pelivaiheessa on tarkoitus tehdä. Peliin suunnitellaan lisättäväksi tutoriaaleja sellaisiin kohtiin, joissa pelaaja kohtaa uuden tilanteen.

Ongintaseikkailun viihdyttävyyttä on tarkoitus parantaa kehittämällä hahmojen persoonallisuutta. Maailmasta ja hahmoista pyritään tekemään entistä

dynaamisempi ja ympäristöön lisätään hauskuuttavia animaatioita, esimerkiksi naurava hylje. Näillä lisäominaisuuksilla ei ole vaikutusta pelimekaniikkaan, vaan niiden tarkoitus on saada pelaajat kiintymään peliin ja sen hahmoihin.

Kosketusnäyttöohjausta tullaan testaamaan lisää potentiaalisten asiakkaiden kanssa ja ohjausmekaniikkaa hiotaan heiltä saatavan palautteen avulla. Näin rakennetaan pelin seuraava versio, jossa käyttäjän kannalta suurimmat ongelmat on korjattu. Tätä versiota testataan edellisen tapaan. Syklin tuloksena päästään asteittain yhä lähemmäs toimivaa kosketusnäyttöohjausta.

7 YHTEENVETO

Kosketusnäyttöjen havaittiin soveltuvan mobiilipelien ohjaimeksi hyvin. Kosketusnäyttöohjauksessa on tiettyjä rajoituksia, jotka on syytä ottaa huomioon pelin kehitysvaiheessa. Suurimpia haasteita ovat ohjauksen tarkkuus, vähäinen taktiili palaute ja painikkeiden käytännöllinen sijoittelu ruudulla. Oikein toteutettuna kosketusnäyttöohjaus on kuitenkin intuitiivinen ja helppo tapa ohjata mobiilipeliä.

Pelimoottorivertailu paljasti että markkinoilta löytyy runsaasti Android-laitteille sopivia pelimoottoreita ja jokainen vertailtu moottori tuki kosketusnäyttöohjausta. Moottorivertailusta oli hyötyä paitsi PenQ Rocketin pelimoottorin valinnassa, myös mahdollisten tulevien peliprojektien kannalta. Työn aikataulun puitteissa ei ollut mahdollista tutkia monimutkaisempien eleiden käyttöä Unityn pelinkehitysympäristössä. Jatkokehityksen kannalta tämä on varmasti yksi alue, jota kannattaa tutkia syvemmin.

Kosketusnäyttöjen ja mobiililaitteiden teorian tutkiminen syvensi osaamistani näiden laitteiden suhteen, mikä mahdollisti PenQ Rocket -projektin toteuttamisen varsin nopealla aikataululla. Projektissa onnistuttiin luomaan prototyyppiversio kosketusnäyttöohjattavasta pelistä, tuomaan se Android-mobiililaitteelle ja testaamaan sitä alustavasti. PenQ Rocket on kuitenkin vielä kaukana valmiista pelistä ja sitä tullaan jatkokehittämään grafiikan, pelattavuuden ja toiminnallisuuksien suhteen.

LÄHTEET

[1] Hartson, R. & Hix, D., *Advances in Human-Computer Interaction*, 3. uudistettu painos. USA: Ablex Publishing, 1992, s.1-33.

[2] Halstead, C., ”Touch screen technology grows popular”, *Artesian Herald*, December 17, 2009. Saatavilla: http://msdadmin.scican.net/mhs/mhs_area_artesian%20herald/Artesian%20Herald%20Volumes/20092010/issue_6/MHSA12-BW-1217.pdf

[3] TimeRime, ”History of Touch Screens and Touch Sensitive Pads”, [www-dokumentti]. Saatavilla: http://timerime.com/en/event/942885/Touch+Screen+Technology+Described+EA+Johnson/ (luettu: 27.5.2014).

[4] Columbia University in the City of New York, ”Computing History / Hewlett-Packard-150”, [www-dokumentti]. Saatavilla: http://www.columbia.edu/cu/computinghistory/hp150.html (luettu: 26.5.2014).

[5] Wikipedia, ”Touchscreen of the HP Series 100 HP-150”, [jpg-kuvatiedosto]. Saatavilla: http://en.wikipedia.org/wiki/File:Hp150_touchscreen_20081129.jpg (luettu 27.5.2014).

[6] Lee, SK.; Buxton, W. & Smith K. C., ”A multi-touch three dimensional touch-sensitive tablet”, *Chi. '85 Proceedings*, (CHI 85), University of Toronto, s. 21-25. Saatavilla: http://www.billbuxton.com/leebuxtonsmith.pdf

[7] Sears, A.; Plaisant, C. & Schneiderman, B., ”A new era for touchscreen applications: High precision, dragging icons, and refined feedback”, *1990 Tech Report*, (CS-TR-2487), University of Maryland, s. 1-24. Saatavilla: http://hcil2.cs.umd.edu/trs/90-01/90-01.pdf

[8] Larry K. Baxter, *Capacitive Sensors: Design and Applications*. USA: Wiley-IEEE Press, 1996. s. 138. ISBN: 978-0-7803-5351-0

[9] Downs, R. 2005., ”Using resistive touch screens for human/machine interface”, *Analog Applications Journal*, (AAJ Q3 2005), Texas Instruments Incorporated, s. 5-9. Saatavilla: http://www.ti.com/lit/an/slyt209a/slyt209a.pdf

[10] Ovaska, S., ”Suorakäyttöä pöytäpinnoilla ja muuten”, [pdf-dokumentti]. Saatavilla: http://www.uta.fi/sis/tie/ui/syksy2012/Luennot/7_postWIMP_6.pdf (luettu: 29.5.2014).

[11] Wroblewski, L., ”Touch Gesture Reference Guide”, [www-dokumentti]. Saatavilla: http://www.lukew.com/ff/entry.asp?1071 (luettu 4.6.2014).

[12] Huang, KY., ”Challenges in Human-Computer Interaction Design for Mobile Devices”, *World Congress on Engineering and Computer Science 2009 Vol. I*, (WCECS 2009), Nova Southeastern University. ISBN: 978-988-17012-6-8 Saatavilla: http://www.iaeng.org/publication/WCECS2009/WCECS2009_pp236-241.pdf

[13] Vatanen, P. & Saastamoinen, A., ”Älypuhelinten yleisimmät viat: näytön rikkoutuminen ja kastuminen”, [www-dokumentti]. Saatavilla: http://yle.fi/aihe/artikkeli/2014/02/13/alypuhelinten-yleisimmat-viat-nayton-rikkoutuminen-ja-kastuminen (luettu: 27.5.2014).

[14] Epaper Central, ”E-paper Technologies Reference”, [www-dokumentti]. Saatavilla: http://www.epapercentral.com/epaper-technologies-guide (luettu 27.5.2014).

[15] Microsoft Research, ”Real-Time Human Pose Recognition in Parts from Single Depth Images”, [pdf-dokumentti]. Saatavilla: http://research.microsoft.com/pubs/145347/bodypartrecognition.pdf (luettu: 27.5.2014).

[16] Tactus Technology, ”A New Dimension of Touch”, [pdf-dokumentti]. Saatavilla: http://tactustechnology.com/wp-content/uploads/2013/09/Tactus_Technology_White_Paper.pdf (luettu 27.5.2014).

[17] Tilastokeskus, ”Internetin käyttö muualla kuin kotona tai työpaikalla”, [www-dokumentti]. Saatavilla: http://www.stat.fi/til/sutivi/2012/sutivi_2012_2012-11-07_kat_003_fi.html (luettu 27.5.2014).

[18] Fingas, J., ”Android tops 81 percent of smartphone market share in Q3”, [www-dokumentti]. Saatavilla: http://www.engadget.com/2013/10/31/strategy-analytics-q3-2013-phone-share/ (luettu 27.5.2014).

[19] Tablet PC Comparison, ”Android Tablets”, [www-dokumentti]. Saatavilla: http://www.tabletpccomparison.net/ (luettu 27.5.2014).

[20] Android Developers, ”Supporting Multiple Screens”, [www-dokumentti]. Saatavilla: http://developer.android.com/guide/practices/screens_support.html (luettu 27.5.2014).

[21] NVIDIA, ”NVIDIA Tegra K1: A New Era in Mobile Computing”, [pdf-dokumentti]. Saatavilla: http://www.nvidia.com/content/PDF/tegra_white_papers/tegra-K1-whitepaper.pdf (luettu 27.5.2014).

[22] Qualcomm, ”Snapdragon S4 Processors: System on Chip Solutions for a New Mobile Age”, [pdf-dokumentti]. Saatavilla: http://www.qualcomm.com/media/documents/files/snapdragon-s4-processors-system-on-chip-solutions-for-a-new-mobile-age.pdf (luettu 27.5.2014).

[23] Android Developers, ”Connectivity”, [www-dokumentti]. Saatavilla: http://developer.android.com/guide/topics/connectivity/index.html (luettu: 4.6.2014).

[24] Lee, A., ”15 essential mobile game development tools”, [www-dokumentti]. Saatavilla: http://www.develop-online.net/tools-and-tech/15-essential-mobile-game-development-tools/0184480 (luettu 27.5.2014).

[25] Totura, N., ”Game Engines For Android”, [www-dokumentti]. Saatavilla: https://software.intel.com/en-us/android/blogs/2012/03/13/game-engines-for-android (luettu 27.5.2014).

[26] Reynolds, C., ”Top IOS & Android Mobile Game Development Tools, Frameworks, Engines & Resources”, [www-dokumentti]. Saatavilla: http://www.mobyaffiliates.com/blog/ios-android-mobile-game-development-tools-frameworks-engines-resources/ (luettu 27.5.2014).

[27] AndEngine, ”Android Game Engine”, [www-dokumentti]. Saatavilla: http://www.andengine.org/blog/ (luettu 27.5.2014).

[28] YoYo Games, ”GameMaker: Studio”, [www-dokumentti]. Saatavilla: https://www.yoyogames.com/studio (luettu 27.5.2014).

[29] YoYo Games, ”Reference Documents: Mouse, Keyboard and Other Controls”, [www-dokumentti]. Saatavilla: http://docs.yoyogames.com/source/dadiospice/002_reference/mouse,%20keyboard%20and%20other%20controls/index.html (luettu 27.5.2014).

[30] GameSalad, ”GameSalad Creator”, [www-dokumentti]. Saatavilla: http://gamesalad.com/creator (luettu 27.5.2014).

[31] Corona Labs, ”Corona SKD”, [www-dokumentti]. Saatavilla: http://coronalabs.com/products/corona-sdk/ (luettu 27.5.2014).

[32] Corona Labs, ”Docs: Tap / Touch / Multitouch”, [www-dokumentti]. Saatavilla: http://docs.coronalabs.com/guide/events/detectEvents/index.html (luettu 27.5.2014).

[33] Marmalade, ”Marmalade SDK”, [www-dokumentti]. Saatavilla: https://www.madewithmarmalade.com/products/marmalade-sdk (luettu 27.5.2014).

[34] Marmalade, ”Marmalade Documentation: Detecting touch, key and accelerometer inputs”, [www-dokumentti]. Saatavilla: http://docs.madewithmarmalade.com/display/MD/Detecting+touch, +key+and+accelerometer+inputs (luettu 27.5.2014)

[35] Unity Technologies, ”Unity – Game Engine”, [www-dokumentti]. Saatavilla: https://unity3d.com/ (luettu 27.5.2014).

[36] Unity Technologies, ”Unity Manual: Mobile Input”, [www-dokumentti]. Saatavilla: http://docs.unity3d.com/Manual/MobileInput.html (luettu 4.6.2014).

[37] Epic Games, ”Unreal Engine: Unreal Development Kit”, [www-dokumentti]. Saatavilla: https://www.unrealengine.com/products/udk/ (luettu 27.5.2014).

[38] Unreal Developer Network, ”Unreal Engine 3 – Mobile Input System”, [www-dokumentti]. Saatavilla: http://udn.epicgames.com/Three/MobileInputSystem.html (luettu 27.5.2014).

[39] Ux Movement, ”Finger Friendly Design: Ideal Mobile Touch Target Sizes”, Saatavilla: http://uxmovement.com/mobile/finger-friendly-design-ideal-mobile-touch-target-sizes/ (luettu: 31.5.2014).

Ohjelmakoodi virtuaaliohjaimen lisäämiseksi AndEngineen

```
package org.anddev.andengine.examples;

import javax.microedition.khronos.opengles.GL10;

import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.engine.camera.hud.controls.BaseOnScreenControl;
import org.anddev.andengine.engine.camera.hud.controls.BaseOnScreenControl.IOnScreenControlListener;
import org.anddev.andengine.engine.camera.hud.controls.DigitalOnScreenControl;
import org.anddev.andengine.engine.handler.physics.PhysicsHandler;
import org.anddev.andengine.engine.options.EngineOptions;
import org.anddev.andengine.engine.options.EngineOptions.ScreenOrientation;
import org.anddev.andengine.engine.options.resolutionpolicy.RatioResolutionPolicy;
import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.entity.scene.background.ColorBackground;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.entity.util.FPSLogger;
import org.anddev.andengine.opengl.texture.TextureOptions;
import org.anddev.andengine.opengl.texture.atlas.bitmap.BitmapTextureAtlas;
import org.anddev.andengine.opengl.texture.atlas.bitmap.BitmapTextureAtlasTextureRegionFactory;
import org.anddev.andengine.opengl.texture.region.TextureRegion;

import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;

/**
 * (c) 2010 Nicolas Gramlich
 * (c) 2011 Zynga
 *
 * @author Nicolas Gramlich
 * @since 00:06:23 - 11.07.2010
 */
public class DigitalOnScreenControlExample extends BaseExample {
	// ===========================================================
	// Constants
	// ===========================================================

	private static final int CAMERA_WIDTH = 480;
	private static final int CAMERA_HEIGHT = 320;
	private static final int DIALOG_ALLOWDIAGONAL_ID = 0;

	// ===========================================================
	// Fields
	// ===========================================================

	private Camera mCamera;

	private BitmapTextureAtlas mBitmapTextureAtlas;
	private TextureRegion mFaceTextureRegion;

	private BitmapTextureAtlas mOnScreenControlTexture;
	private TextureRegion mOnScreenControlBaseTextureRegion;
	private TextureRegion mOnScreenControlKnobTextureRegion;

	private DigitalOnScreenControl mDigitalOnScreenControl;

	// ===========================================================
	// Constructors
	// ===========================================================

	// ===========================================================
	// Getter & Setter
	// ===========================================================

	// ===========================================================
	// Methods for/from SuperClass/Interfaces
	// ===========================================================

	@Override
	public Engine onLoadEngine() {
		this.mCamera = new Camera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);
		return new Engine(new EngineOptions(true, ScreenOrientation.LANDSCAPE, new RatioResolutionPolicy(CAMERA_WIDTH, CAMERA_HEIGHT), this.mCamera));
	}

	@Override
	public void onLoadResources() {
		BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/");

		this.mBitmapTextureAtlas = new BitmapTextureAtlas(32, 32, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
		this.mFaceTextureRegion = BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.mBitmapTextureAtlas, this, "face_box.png", 0, 0);

		this.mOnScreenControlTexture = new BitmapTextureAtlas(256, 128, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
		this.mOnScreenControlBaseTextureRegion = BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.mOnScreenControlTexture, this, "onscreen_control_base.png", 0, 0);
		this.mOnScreenControlKnobTextureRegion = BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.mOnScreenControlTexture, this, "onscreen_control_knob.png", 128, 0);

		this.mEngine.getTextureManager().loadTextures(this.mBitmapTextureAtlas, this.mOnScreenControlTexture);
	}

	@Override
	public Scene onLoadScene() {
		this.mEngine.registerUpdateHandler(new FPSLogger());

		final Scene scene = new Scene();
		scene.setBackground(new ColorBackground(0.09804f, 0.6274f, 0.8784f));

		final int centerX = (CAMERA_WIDTH - this.mFaceTextureRegion.getWidth()) / 2;
		final int centerY = (CAMERA_HEIGHT - this.mFaceTextureRegion.getHeight()) / 2;
		final Sprite face = new Sprite(centerX, centerY, this.mFaceTextureRegion);
		final PhysicsHandler physicsHandler = new PhysicsHandler(face);
		face.registerUpdateHandler(physicsHandler);

		scene.attachChild(face);

		this.mDigitalOnScreenControl = new DigitalOnScreenControl(0, CAMERA_HEIGHT - this.mOnScreenControlBaseTextureRegion.getHeight(), this.mCamera,
		this.mOnScreenControlBaseTextureRegion, this.mOnScreenControlKnobTextureRegion, 0.1f, new IOnScreenControlListener() {
			@Override
			public void onControlChange(final BaseOnScreenControl pBaseOnScreenControl, final float pValueX, final float pValueY) {
				physicsHandler.setVelocity(pValueX * 100, pValueY * 100);
			}
		});
		this.mDigitalOnScreenControl.getControlBase().setBlendFunction(GL10.GL_SRC_ALPHA, GL10.GL_ONE_MINUS_SRC_ALPHA);
		this.mDigitalOnScreenControl.getControlBase().setAlpha(0.5f);
		this.mDigitalOnScreenControl.getControlBase().setScaleCenter(0, 128);
		this.mDigitalOnScreenControl.getControlBase().setScale(1.25f);
		this.mDigitalOnScreenControl.getControlKnob().setScale(1.25f);
		this.mDigitalOnScreenControl.refreshControlKnobPosition();

		scene.setChildScene(this.mDigitalOnScreenControl);

		return scene;
	}
```

```
        @Override
        public void onLoadComplete() {
                this.showDialog(DIALOG_ALLOWDIAGONAL_ID);
        }

        @Override
        protected Dialog onCreateDialog(final int pID) {
                switch(pID) {
                        case DIALOG_ALLOWDIAGONAL_ID:
                                return new AlertDialog.Builder(this)
                                .setTitle("Setup...")
                                .setMessage("Do you wish to allow diagonal directions on the OnScreenControl?")
                                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                                        @Override
                                        public void onClick(final DialogInterface pDialog, final int pWhich) {
                                                DigitalOnScreenControlExample.this.mDigitalOnScreenControl.setAllowDiagonal(true);
                                        }
                                })
                                .setNegativeButton("No", new DialogInterface.OnClickListener() {
                                        @Override
                                        public void onClick(final DialogInterface pDialog, final int pWhich) {
                                                DigitalOnScreenControlExample.this.mDigitalOnScreenControl.setAllowDiagonal(false);
                                        }
                                })
                                .create();
                }
                return super.onCreateDialog(pID);
        }

        // ===========================================================
        // Methods
        // ===========================================================

        // ===========================================================
        // Inner and Anonymous Classes
        // ===========================================================
}
```

Aluesiirtymän ohjelmakoodi

```
using UnityEngine;
using System.Collections;

public class EnterArea : MonoBehaviour {

    public string targetArea;

    void OnMouseDown() {
        Application.LoadLevel (targetArea);
    }
}
```

Inventaarion ohjelmakoodi

```
using UnityEngine;
using System.Collections;

public class Inventory : MonoBehaviour {

    public GameObject[] fishAmounts;
    public GameObject[] eggAmounts;

    private bool inventoryOpen=false;

    // Use this for initialization
    void Start () {
        UpdateAmounts ();
    }

    void OnMouseDown()
    {
        //toggle inventory visibility
        if(inventoryOpen){
            inventoryOpen = false;
            animation.Play ("InventoryClose");
        } else {
            this.UpdateAmounts();
            inventoryOpen = true;
            animation.Play ("InventoryOpen");
        }
    }

    void UpdateAmounts() {
        //refreshing fish numbers in inventory
        for (int x=0;x<fishAmounts.Length;x++) {
            fishAmounts[x].GetComponent<TextMesh>().text = GameStatus.amountFish[x].ToString();
        }

        for (int x=0;x<eggAmounts.Length;x++) {
            eggAmounts[x].GetComponent<TextMesh>().text = GameStatus.amountEggs[x].ToString();
        }

    }
}
```